DISEÑO E IMPLEMENTACIÓN DE UN BACKEND PARA LA GESTIÓN Y REALIZACIÓN DE PRÁCTICAS REMOTAS DE LABORATORIO

INGENIERÍA TÉCNICA DE TELECOMUNICACIONES: TELEMÁTICA

AUTOR: ROBERTO SANTIAGO DÉVORA

TUTOR: JAVIER FERNÁNDEZ MUÑOZ

A todos aquellos que siempre me animaron a terminar y a no dejar las cosas a medias.

A Nieves, por demostrarme que siempre se puede.

A Mari Ángeles y Asurio y su fe sin límites.

Y en especial a Ana, pues ella es el inicio y el fin de todo.

"Si abordas una situación como un asunto de vida o muerte, morirás muchas veces"

Adam Smith

Tabla de contenido

Tabla de figuras

Tabla de tablas

Resumen

El objetivo de este proyecto es desarrollar un sistema centralizado para la gestión y reserva de dispositivos ubicados en los laboratorios de la Universidad, así como dar la posibilidad de realizar prácticas sobre los mismos, de manera remota, por parte de los alumnos. El sistema a desarrollar será un conjunto de servicios web que permitan a terceras aplicaciones (aplicación web, aplicación nativa para dispositivo móvil, etc…) interaccionar con el sistema centralizado de gestión para así mostrar los datos a los usuarios finales, y además servir como puente en la comunicación entre dichas aplicaciones/usuarios y los dispositivos donde se realizarán las prácticas remotas. En este documento se van a detallar las distintas fases del proyecto, así como sus componentes y funcionamiento.

Abstract

The objective of this project is the developing of a centralized system to manage and reserve the devices located in the University laboratories, besides giving the possibility to execute practices with them, in a remote mode, by the students. The system will be a group of web services that allows to third applications (web applications, native mobile device application, etc.) interact with the management centralized system to show the information to the final users and, also be a bridge for the communication between those applications/users and the devices where the remote practice will have place. In this document the different phases of the project will be detailed as well as their components and functioning.

1 Introducción

1.1 Introducción general

Cada vez es más frecuente que no tengamos que estar frente a un recurso para poder hacer uso del mismo: cámaras de vigilancia, termostatos, persianas… todos ellos accesibles a través de Internet, lo que se conoce actualmente como Internet of Things.

Este ambicioso mundo de interconectarse con prácticamente cualquier dispositivo nos lleva a pensar que gran parte de las tareas que realizamos en nuestro día a día la podríamos realizar desde cualquier otro punto y además acceder desde diferentes medios: ordenadores, tablets, smartphones…

Dentro de este concepto se mueve la Universidad Carlos III de Madrid, que pretender conseguir que cualquier trámite puede ser realizado a distancia: acceso al material didáctico, trámites burocráticos, solicitud de libros, expediente académico… y por supuesto también la realización de prácticas. Esto es ya un hecho en su mayoría, pero hay ciertas trabas cuando lo que se desea es trabajar con un dispositivo físico: motores, circuitos, autómatas… ya que hasta el momento es necesario estar frente al dispositivo para poder interactuar con él.

Lo que se pretende con este proyecto es dar la posibilidad a los alumnos de realizar, de forma remota, una reserva concreta sobre un cierto número de dispositivos disponibles en los laboratorios de la Universidad para que a posteriori pueda realizar una práctica con ellos como si estuvieran sentados frente a él. Para ello se desarrollará un motor de gestión y reservas de dispositivos que sea accesible por terceras aplicaciones, de forma que existan diferentes aplicaciones adaptadas al medio de acceso del usuario final.

1.2 Motivación del proyecto

Como se acaba de describir, la Universidad Carlos III de Madrid busca poder dar la facilidad a sus alumnos de realizar las prácticas universitarias de forma remota. Con ello se consigue, no solo darles la posibilidad de probar sus implementaciones antes de asistir a los talleres de forma física sino también poder acceder a los dispositivos existentes a horas en las que no sería posible debido al horario del recinto, y por tanto se aumenta la disponibilidad de los laboratorios.

Además, tanto por parte de la Universidad como mi parte, se busca perseguir un reto tecnológico a la hora de trabajar con tecnologías punteras las cuales sean de las más demandadas en el mercado actual de trabajo. Es por ello que se decide enfocar este proyecto como un sistema de interconexión, que permita la operación de un

conjunto externo de sistemas, sirviendo este como núcleo y nexo comunicador para la gestión y reserva de dispositivos físicos de los laboratorios.

1.3 Objetivos del proyecto

El objetivo del proyecto es el desarrollo e implementación de un conjunto de servicios que permita la gestión y reserva de los dispositivos habilitados para la realización de prácticas remotas. Además, el sistema tiene que ser centralizador de terceras aplicaciones, que permitan una gestión común para las distintas aplicaciones desde las cuales un alumno podrá acceder al sistema.

Los puntos principales son los siguientes:

- Sistema centralizado de gestión de dispositivos y reservas, y que las aplicaciones finales se desentiendan de la complejidad existente bajo la capa de gestión, además de permitir que todas ellas compartan los mismos datos.

- Control de acceso a las aplicaciones que harán uso del sistema, para que solo aplicaciones/usuarios autorizados puedan acceder a los datos.

- Control sobre los dispositivos finales, permitiendo el arranque y la parada de los mismos, de forma que no se haga uno abuso de ellos.

- Personalización y gestión de los distintos tipos de prácticas, permitiendo así una múltiple configuración dentro de cada dispositivo.

2 Estado del arte

En este apartado se van a explorar las distintas maneras existentes mediante las cuales se puede realizar la interconexión de sistemas, y en especial de como el mundo de las APIs REST ha marcado el rumbo actual para el intercambio de información y comunicación entre distintas aplicaciones de software.

Además, contaremos como la introducción de sistemas de autenticación delegada han ayudado a crear de forma más rápida nuevos desarrollos que independizan a las nuevas aplicaciones de crear sistemas de autorización y autenticación en sus entornos, dejando dicha tarea a herramientas con una mayor conocimiento del entorno.

2.1 Evolución de la conectividad entre sistemas: la llegada de las APIs REST

Desde que existe el software y las herramientas para desarrollar nuevas aplicaciones se trabajaba en la creación de sistemas capaces de hacer por ellos mismos todo el trabajo que tenían que llevar a cabo, sistemas muy completos capaces de realizar tareas muy diversas. Sin embargo, con la llegada de las redes y la interconexión de equipos y dispositivos, se vio que había un gran potencial en crear sistemas más especializados en tareas concretas, que podían delegar en terceras aplicaciones las tareas que no quedaban dentro de su ámbito de trabajo.

Esta interconexión se realiza mediante la llamada a procedimientos, métodos o servicios y debe de asumir lo siguiente:

- Comunicación de una aplicación a otra
- Sin necesidad de conocer
    - Lenguaje en el cual está escrita la otra aplicación
    - Equipo en el cual se ejecuta la otra aplicación

De tal forma, el esquema de interconexión entre aplicaciones quedaría de la siguiente manera:

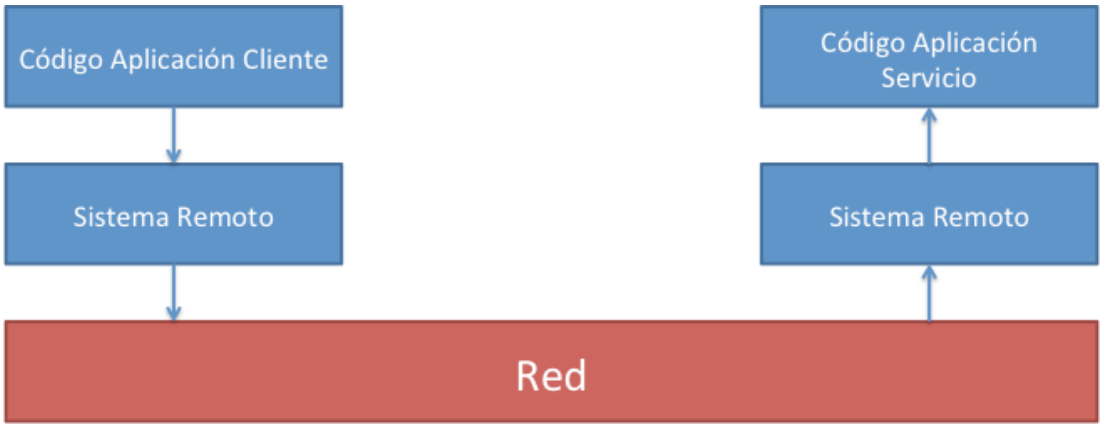


Figura 1: Esquema de interconexión de aplicaciones

Además, hay una serie de características que han de quedar definidas a la hora de realizar un conjunto de servicios remotos y son las siguientes:

- **Descripción:** información sobre las acciones que realiza el servicio
- **Presentación:** modelo de petición/respuesta
- **Codificación:** esquema definido para el paso de información
- **Protocolo:** definición de los metadatos por los que se va a mover la información
- **Transporte:** enlace entre cliente y servicio
- **Seguridad:** capa de autenticación, autorización y encriptación
- **Sesión:** estado de la comunicación entre cliente y servicio
- **Directorio:** información sobre cómo encontrar el servicio

A continuación vamos a realizar un repaso de la evolución de estos sistemas de interconexión y las características que aporta cada uno de ellos para así comprender el motivo por el cual se ha decidido realizar el proyecto como un API REST.

Llamada a procedimientos remotos: CORBA y RMI

CORBA nace en 1991 para tratar de solucionar un problema existente por entonces, comunicar dos sistemas software completamente diferentes: sistemas operativos, lenguajes de programación, hardware, paradigmas... Para ello, CORBA propone el uso de un lenguaje de definición de interfaces (IDL) que permita especificar la representación de los modelos de cara al mundo exterior(Stub/Skeleton).

Posteriormente, dicha interfaz es transformada mediante un mapeo a un lenguaje común que será usado en la capa de transporte entre cliente y servicio (ORB), de tal forma que sea posible realizar una comunicación común entre los extremos del mensaje, y que esta vuelva a ser transformada en el mapeo declarado en el lado del servicio.

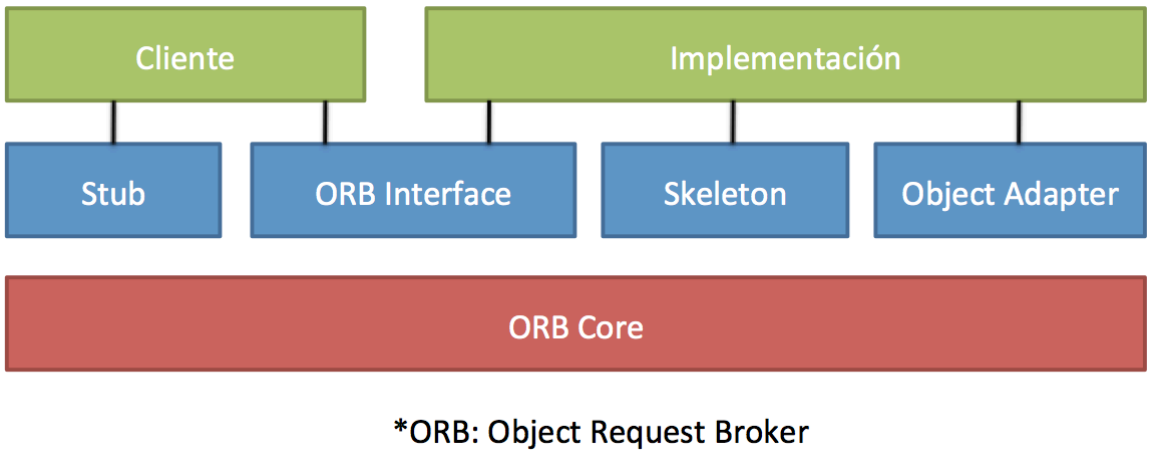


Figura 2: Arquitectura CORBA

Los beneficios que aporta CORBA son los siguientes:

- Independencia de lenguaje, tecnología y sistema operativo
- Transacciones atómicas
- Tipado de los datos, que reduce el error humano al identificar el tipo de dato esperado en el IDL. Además, existe la libertada de declarar un dato como tipo "ANY" pudiendo introducir cualquier valor.
- Libertad en la transferencia de los datos, donde toda la capa de envío queda bajo la responsabilidad del ORB que provee mecanismos para el tratamiento de errores y excepciones.
- Compresión de los datos, ya que el ancho de banda en un bien escaso, y más en aquellos años, los datos son transformados a formato binario con la posibilidad de compresión y descompresión entre los extremos.

Si hablamos de RMI, podemos decir que estamos tratando con la simplificación CORBA en el mundo Java. Esto quiere decir que si lo que queremos es comunicar un cliente/servicio en el que ambos están desarrollados en Java, se puede hacer uso de Java RMI en lugar de crear el desarrollo para que se adapte a CORBA. La definición del par Stub/Skeleton se sigue manteniendo, pero para la aplicación cliente es como si estuviera ejecutando todo su código de manera local, además de que usará los servicios de "lookup" para encontrar el conjunto de servicios remotos a los que tiene que llamar.

Como principales ventajas cabría destacar la facilidad de uso e implementación, que el paso de objetos siempre se hará por referencia y el sistema de "Garbage Collector" distribuido.

Arquitectura orientada a servicios: SOAP

Cuando hablamos de SOAP, estamos hablando de un protocolo para el intercambio de información estructurada cuyo implementación se basa en un conjunto de mensajes con formato XML. Este protocolo tiene tres características principales:

- **Extensibilidad:** junto a su definición base se van a acoplando nuevas extensiones que enriquecen el protocolo, como pueden ser la seguridad o el routing.
- **Neutralidad:** puede ser utilizado con cualquier protocolo de transporte (TCP, UDP, JMS…) aunque mayoritariamente se utiliza HTTP o SMTP
- **Independencia:** permite cualquier modelo de programación

Como hemos indicado, el mensaje de intercambio entre cliente y servicio siempre será en formato XML, cumpliendo una estructura definida en el protocolo:

- **Envelope:** la parte del mensaje que lo identificará como un mensaje SOAP (Obligatorio)
- **Header:** permite asociar información relativa a cómo debe de ser tratado el mensaje en destino
- **Body:** la información relativa a la petición/respuesta (Obligatorio)
- **Fault:** la información relativa a los errores que se hayan producido durante el envío y procesado del mensaje

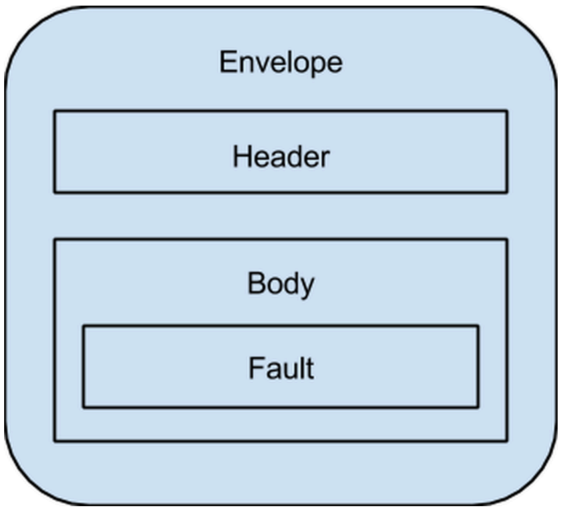


Figura 3: Arquitectura orientada a servicios - SOAP

A continuación se puede ver un ejemplo de mensaje SOAP donde se aprecia el detalle en especificación que se realiza del formato del mensaje, pero a su vez lo cargado que queda con etiquetas de apertura y cierre en todos los elementos.

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

Fuente: http://www.w3schools.com/xml/xml_soap.asp

Para la descripción del servicio y el formato del intercambio de los mensajes se utiliza el estándar WSDL (Web Service Description Language), de esta forma un cliente es capaz de conectar con la especificación del conjunto de servicios publicados y obtener qué servicios están desplegados y cómo serán los mensajes que se esperan y el tipado de los mismos.

A continuación se muestra un ejemplo de un fichero WSDL para la especificación de un conjunto de servicios web.

```
<?xml version="1.0"?>

<wsdl:definitions name="EndorsementSearch"
 targetNamespace="http://namespaces.snowboard-info.com"
 xmlns:es="http://www.snowboard-info.com/EndorsementSearch.wsdl"
 xmlns:esxsd="http://schemas.snowboard-info.com/EndorsementSearch.xsd"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

 <wsdl:types>
   <xsd:schema targetNamespace="http://namespaces.snowboard-info.com"
     xmlns:xsd="http://www.w3.org/1999/XMLSchema">
     <xsd:element name="GetEndorsingBoarder">
       <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="manufacturer" type="string"/>
           <xsd:element name="model" type="string"/>
        </xsd:sequence>
       </xsd:complexType>
     </xsd:element>
   </xsd:schema>
 </wsdl:types>


 <wsdl:message name="GetEndorsingBoarderRequest">
   <wsdl:part name="body" element="esxsd:GetEndorsingBoarder"/>
 </wsdl:message>

 <wsdl:portType name="GetEndorsingBoarderPortType">

   <wsdl:operation name="GetEndorsingBoarder">
     <wsdl:input message="es:GetEndorsingBoarderRequest"/>
   </wsdl:operation>
 </wsdl:portType>

 <wsdl:binding name="EndorsementSearchSoapBinding"
               type="es:GetEndorsingBoarderPortType">
   <soap:binding style="document"
                 transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="GetEndorsingBoarder">
     <soap:operation soapAction="http://www.snowboard-info.com/EndorsementSearch"/>
   <wsdl:input>
       <soap:body use="literal"
                  namespace="http://schemas.snowboard-info.com/EndorsementSearch.xsd"/>
     </wsdl:input>
   </wsdl:operation>
 </wsdl:binding>

 <wsdl:service name="EndorsementSearchService">
   <wsdl:documentation>snowboarding-info.com Endorsement Service</wsdl:documentation>
   <wsdl:port name="GetEndorsingBoarderPort"
              binding="es:EndorsementSearchSoapBinding">
     <soap:address location="http://www.snowboard-info.com/EndorsementSearch"/>
   </wsdl:port>
 </wsdl:service>

</wsdl:definitions>
```

Fuente: http://www.w3.org/2001/03/14-annotated-WSDL-examples.html

Como resumen del uso de SOAP como estándar para la interconexión entre cliente y servicio vamos a ver sus ventajas e inconvenientes.

Ventajas

- Gran interoperabilidad gracias al uso de XML como formato del mensaje
- Fácilmente escalable
- Sin problemas en la capa de transporte
- Implementado en cualquier lenguaje de desarrollo y plataforma
- Puede ser utilizado como usuario anónimo o con autenticación, gracias a sus módulos de extensión como WS-Trust y WS-Security

Inconvenientes

- Mensajes sobrecargados debido a la naturaleza del formato XML
- Depende de WSDL

El mundo HTTP: Servicios RESTFul

Llegados a este punto no encontramos con la arquitectura de servicios web de tipo REST. Esta se basa en el intercambio de mensajes bajo el protocolo de comunicaciones HTTP, de forma que se le da mucha más importancia al contenido del mensaje en sí mismo que al formato especificado, ya que esto queda resuelto mediante el estándar HTTP y el uso de sus verbos.

Las características más importantes de los servicios de tipo RESTFul son las siguientes:

- **Modelo cliente/servidor sin estado:** en cada uno de los mensajes viaja la información necesaria para comprender la petición, sin necesidad de recordar el estado de la comunicación en la cadena de mensajes.
- **Cacheable:** tanto en el cliente como en pasos intermedios cabe la posibilidad de cachear los resultados para una misma petición, sin tener que llegar al otro extremo para saber el resultado de la misma. Esta característica mejora tanto la escalabilidad como el rendimiento de la aplicación consumidora de los servicios.
- **Conjunto de operaciones definidas:** HTTP cuenta desde sus comienzos con un listado de acciones útiles para todos sus recursos (GET, POST, PUT, DELETE…) y este mismo es utilizado en los servicios REST para definir las distintas opciones que es capaz de realizar un "endpoint" o servicio.
- **Identificador de recursos** mediante URIs únicas.
- **Variedad en el formato de respuesta del mensaje:** al tratarse de HTTP, el tipo de formato del mensaje se indica en la cabecera del mismo para permitir al receptor decodificar el mismo, permitiendo así una gran variedad de formatos en la respuesta (XML, HTML, JSON, TXT…)

De esta forma el esquema de comunicación cliente/servicio con el estándar RESTFul queda de la siguiente manera:

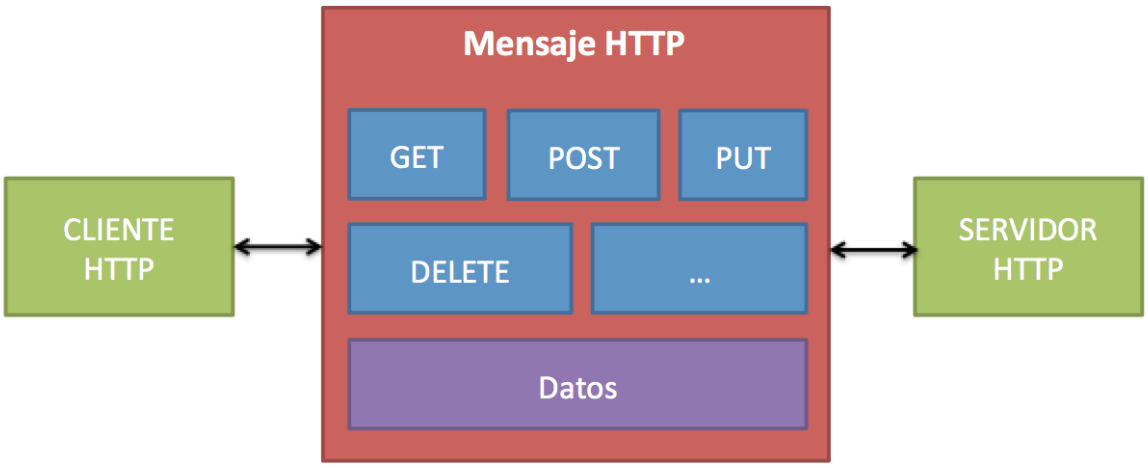


Figura 4: Comunicación HTTP sin estado

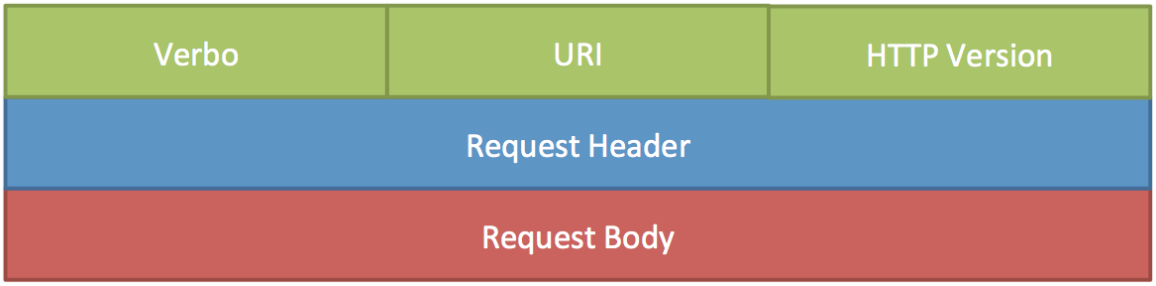


Figura 5: HTTP Request

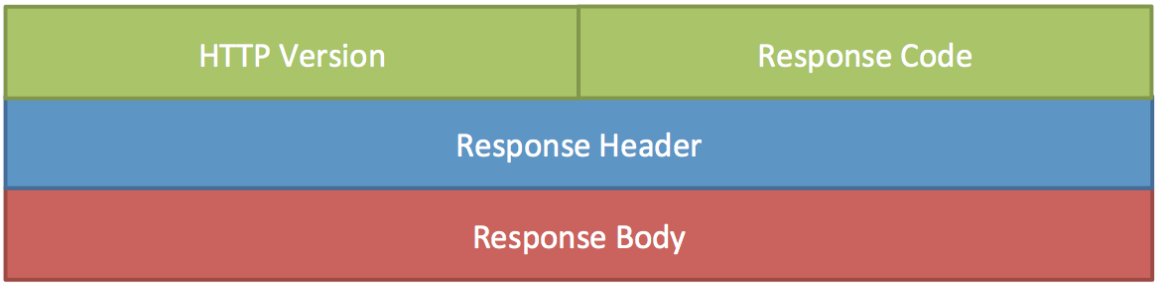


Figura 6: HTTP Response

En todo ese intercambio de información, como se ha indicado, se centra el foco en el cuerpo del mensaje (documento) en lugar de especificaciones de formatos, tipificación de los datos, estructura… y por ello el consumo de red se ve reducido drásticamente respecto a los anteriores métodos conocidos.

A continuación se puede ver un ejemplo de petición/respuesta entre un cliente y un servicio donde se comprueba que la mayor parte del mensaje corresponde al cuerpo del mismo:

```
POST http://MyService/Person/
Host: MyService
Content-Type: text/xml; charset=utf-8
Content-Length: 123
<?xml version="1.0" encoding="utf-8"?>
<Person>
 <ID>1</ID>
 <Name>M Vaqqas</Name>
 <Email>m.vaqqas@gmail.com</Email>
 <Country>India</Country>
</Person>
```

Por tanto, y como hemos visto, la comunicación a través de este estándar es mucho más fluida que en las opciones anteriores y nos encontramos con una gran sencillez a la hora de implementar tanto clientes como servicios en ambos extremos. Es por esto por lo que la mayoría de empresas que ofrecen sus servicios a terceros están optando en la actualidad por realizar dichos servicios bajo APIs REST y es por esta misma razón por la que se ha elegido realizar el proyecto en este estándar de comunicación.

Comunicación directa: WebSockets

Además de los que ya conocemos, ahora contamos con un nuevo actor, los WebSockets. Este mecanismo funciona como comunicación directa y bidireccional entre cliente y servicio mediante el protocolo TCP.

No se puede tener en cuenta aún como especificación disponible para la utilización como publicación de servicios, ya que carece de muchas características que le harían útil para tal función. Sin embargo, cabe destacar que es muy utilizado para la comunicación entre aplicaciones clientes que corren un navegador y han de establecer comunicación directa con el servidor para la realización de una tarea concreta y necesitan una comunicación fluida con el otro extremo.

Para finalizar, una vez vistas todas las alternativas individualmente y teniendo en cuenta que los WebSockets no se pueden adoptar como una solución válida, vamos a ver de forma resumida una comparativa de las diferentes formas que tenemos de interconectar dos sistemas.

	CORBA	RMI	SOAP	RESTFul
Descripción	IDL	Java	WSDL	No (Aunque aporta herramientas de especificación como WADL, Swagger o RAML)
Presentación	Mapeo de lenguaje (IDL) y buen tipado de datos	Remoto y serializable	RPC (Remote Procedure Call)	No (Aunque cuenta con frameworks de apoyo como JAX-WS)
Codificación y Protocolo	Binario, difícil de implementar	JRMP	XML y HTTP/SMTP/JMS…	JSON, HTML, XML…
Transporte	Directo	TCP	HTTP, JMS, TCP…	HTTP
Seguridad	Sin encriptación	SSL	WS-Security, WS-Trust, WS-SecureConversation	SSL
Sesión	No	Solo en la capa de transporte (JINI)	WS-ReliableMessaging, WS-AtomicTransactions	Sesión HTTP (mecanismos como las cookies o token de sesión)
Directorio	Nombre del servicio	RMI, el cliente conoce el registro. JINI, el cliente lanza mensaje multicast al servicio de lookup	No	DNS
GCD	No	Si	No	No
Acoplamiento	Alto	Alto	Bajo	Bajo

Tabla 1: Comparativa de los sistemas de interconexión

2.2 Autenticación en los sistemas remotos

Puesto que estamos hablando de crear una aplicación que sea consumida por terceros, esta debería de estar protegida por un sistema de autenticación, autenticación y control de acceso a los servicios (AAA: Authentication, Authorization, Accounting).

No todos nuestros recursos podrán ser públicos, ni todos los recursos que sean privados podrán ser accedidos por todos los consumidores, así que es por ello que

se hace necesario tener un sistema que permita aplicar estos mecanismos de seguridad.

Autenticación

En este paso se comprueba la identidad de la entidad que quiere acceder al sistema. La autenticación se consigue mediante la presentación de una propuesta de identidad, por ejemplo un nombre de usuario, y la demostración de estar en posesión de las credenciales que permiten comprobarla. Ejemplos posibles de estas credenciales son las contraseñas, los testigos de un solo uso (one-time tokens), los Certificados Digitales o los números de teléfono en la identificación de llamadas.

Autorización

Se refiere a la concesión de privilegios específicos (incluyendo "ninguno") a una entidad o usuario basándose en su identidad (autenticada), los privilegios que solicita, y el estado actual del sistema. Las autorizaciones pueden también estar basadas en restricciones, tales como restricciones horarias, sobre la localización de la entidad solicitante, la prohibición de realizar logins múltiples simultáneos del mismo usuario, etc. La mayor parte de las veces el privilegio concedido consiste en el uso de un determinado tipo de servicio.

Control de acceso

El Control o "Accounting" se refiere al seguimiento del consumo de los recursos por los usuarios. Esta información puede usarse posteriormente para la administración, planificación u otros propósitos. La contabilización en tiempo real es aquella en la que los datos generados se entregan al mismo tiempo que se produce el consumo de los recursos. En las aplicaciones este sistema suele delegarse en ficheros generados de los logs de la propia aplicación, que son tratados con posterioridad. La información típica que un proceso de control de acceso registra es la identidad del usuario, el tipo de servicio que se le proporciona y cuando comenzó a usarlo.

Antes de hablar de los sistemas de autenticación/autorización hay que tener en cuenta que el protocolo de comunicación HTTP es un protocolo no securizado y por tanto, toda comunicación en la que se vea involucrada un paso de información de credenciales habrá de estar previamente cifrada. En este caso, HTTP se apoya sobre SSL/TSL (Security Socket Layer / Transport Layer Security) para realizar la comunicación sobre el protocolo HTTPS. El protocolo SSL y todos sus avances se basan en autenticar los dos extremos de una comunicación de red mediante una criptografía asimétrica, de tal forma que ambas partes intercambian sus claves públicas y validan la identidad y autenticidad del otro mediante el uso de los certificados digitales.

Actualmente existen diferentes técnicas para realizar la autenticación en las conexiones HTTP:

Basic Access Authentication

Es la autenticación clásica por usuario/password. Debido a que este mecanismo no provee confidencialidad en el momento de la transmisión (ningún tipo de encriptación) se crea un parámetro asociado a los credenciales que será enviado en la cabecera de todas las peticiones.

Por tanto, una vez que el cliente se ha autenticado correctamente, el servidor le responde con el siguiente parámetro:

```
WWW-Authenticate: Basic realm="nmrs_m7VKmomQ2YM3:"
```

Y las siguientes llamadas del cliente enviarán el siguiente parámetro:

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

OAuth 1.0

Con este estándar de autenticación lo que conseguimos es que una vez que el usuario se ha autenticado con sus credenciales de usuario/password o apiKey/apiSecret desde el servicio se le genera de forma automática un token de verificación que será el que se utilice a partir de ese momento para hacer las llamadas siguientes. Asociado al token se registran los parámetros de timestamp de la llamada y el origen de procedencia de la petición para así asegurar que en ningún caso tenemos una intromisión o suplantación de identidad en la comunicación. Este token de verificación puede tener una fecha de caducidad establecida para forzar a las aplicaciones llamantes a volver a validar sus credenciales.

OAUth 2.0

En este caso sigue la misma filosofía de OAuth 1.0 pero está más enfocado para hacer uso de credenciales de un usuario final por parte de terceras aplicaciones. Es decir, una aplicación necesita acceso a los datos de otra aplicación para un determinado usuario, es por tanto el usuario el que se autentica contra la aplicación final y la redirección del token válido es devuelta a la aplicación origen para que se encargue de almacenar el token válido perteneciente al usuario y pueda hacer llamadas a la aplicación final en su nombre.

Securización física

Además de todos los estándares existentes siempre nos queda la securización de manera física, es decir, una aplicación o servicio sólo puede ser accesible desde determinadas instancias. Podríamos hacer reglas de firewall de entrada al servicio, patrones de llamadas, acceso solo dentro de la misma red… Esta forma de securizar la aplicación es la más fiable y sencilla, pero a su vez es la más restrictiva

ya que limita mucho la posibilidad de que terceras aplicaciones se comuniquen con el servicio de forma abierta.

Como último apunte sobre la securización de servicios, en este caso de tipo API Rest, existen en el mercado herramientas que permiten exponer al mundo este tipo de servicio sin tener que preocuparse por temas como la autenticación y securización, es lo que se conoce como APP-IN-THE-MIDDLE.

Estos aplicaciones/servicios permiten a los proveedores finales de servicios disponer de una serie de herramientas como serían:

- Creación del portal para desarrolladores, donde se muestre la documentación de los distintos servicios que hay publicados y su uso.
- Bloqueo, acceso y autenticación como si fuera un proxy que deriva las llamadas a los servicios finales.
- Monetizar las llamadas a los servicios, generando de esta manera un retorno de la inversión según el uso.
- Datos personalizados sobre el uso de los servicios.

Dentro de este tipo de aplicaciones nos encontramos con las siguientes empresas:

- http://www.mashery.com/
- https://apigee.com
- https://aws.amazon.com/es/api-gateway/
- https://www.mashape.com/
- https://genoa.beeva.com/

3 Análisis

3.1 Introducción

El análisis es la fase del proyecto en el que el analista recoge las necesidades del usuario, genera los casos de usuario y los transforma en requisitos de software.

Es una fase crítica dentro de un proyecto. Un mal análisis lleva al fracaso del proyecto con casi total seguridad. Sin embargo, un buen análisis sienta las bases para poder cubrir las necesidades de el cliente con la solución a implementar.

3.2 Descripción general

El primer paso a dar es recoger todos los requisitos de usuario que deben ser cubiertos por la solución. Estos requisitos pueden ser de capacidad, puesto que recojan funcionalidades que debe cubrir la solución, o de restricción, ya que restrinjan la cobertura de los primeros.

3.2.1 Capacidades generales

A continuación se incluyen los requisitos de usuario identificados durante el análisis.

Crear nueva reserva			
El API debe disponer de un método para poder guardar una reserva de dispositivo para un usuario en una fecha y horario concretos.			
Identificador	RC01	**Prioridad**	Alta

Tabla 2: RC01 - Crear nueva reserva

Modificar reserva			
El API debe disponer de un método para poder modificar una reserva ya creada anteriormente cambiando uno o varios de los valores (fecha, hora, laboratorio y/o dispositivo)			
Identificador	RC02	**Prioridad**	Baja

Tabla 3: RC02 - Modificar reserva

Eliminar reserva			
El API debe disponer de un método para poder eliminar una reserva previamente creada.			
Identificador	RC03	**Prioridad**	Alta

Tabla 4: RC03 - Eliminar reserva

Recuperar detalle reserva			
El API debe disponer de un método que permita recuperar la información al detalle de una reserva no vencida.			
Identificador	RC04	**Prioridad**	Alta

Tabla 5: RC04 - Recuperar detalle reserva

Recuperar reservas de usuario			
El API debe disponer de un método que permita recuperar el listado de todas las reservas nos vencidas de un usuario con su información básica.			
Identificador	RC05	**Prioridad**	Alta

Tabla 6: RC05 - Recuperar reservas de usuario

Buscar laboratorios			
El API debe disponer de un método que permita buscar los laboratorios que estén disponibles en algún momento a partir del momento actual y hasta el periodo configurado.			
Identificador	RC06	**Prioridad**	Alta

Tabla 7: RC06 - Buscar laboratorios

Buscar dispositivos			
El API debe disponer de un método que permita buscar los dispositivos que estén disponibles en algún momento a partir del momento actual y hasta el periodo configurado. Se podrá buscar por diferentes parámetros: tipo, dispositivo concreto y por laboratorio.			
Identificador	RC07	**Prioridad**	Baja

Tabla 8: RC07 - Buscar dispositivos

Recuperar agenda de dispositivo			
El API debe tener un método que permita recuperar los horarios disponibles para un dispositivos concreto.			
Identificador	RC08	**Prioridad**	Alta

Tabla 9: RC08 - Recuperar agenda de dispositivo

Abrir conexión			
El API debe tener un método que devuelva los datos de conexión con el dispositivo de laboratorio y ponga en funcionamiento dicho dispositivo.			
Identificador	RC09	**Prioridad**	Alta

Tabla 10: RC09 - Abrir conexión

Cerrar conexión			
El API debe tener un método que permita enviar al dispositivo de laboratorio la orden de que interrumpa su funcionamiento.			
Identificador	RC10	**Prioridad**	Alta

Tabla 11: RC10 - Cerrar conexión

Creación agenda diaria			
El sistema debe generar a día vencido la agenda para el nuevo día disponible			
Identificador	RC11	**Prioridad**	Media

Tabla 12: RC11 - Creación agenda diaria

Consolidación de reservas pasadas			
El sistema debe realizar un backup de las reservas pasadas a un histórico y eliminar todas las entradas vencidas de la agenda.			
Identificador	RC12	**Prioridad**	Alta

Tabla 13: RC12 - Consolidación de reservas pasadas

Sistema independiente del usuario final			
El sistema debe responder de la misma manera independientemente de quién sea el sistema que le invoque.			
Identificador	RC13	**Prioridad**	Alta

Tabla 14: RC13 - Sistema independiente del usuario final

Sistema de gestión homogéneo y centralizado			
Todas las llamadas al API serán resueltas de forma homogénea atendiendo a sus parámetros y manteniendo siempre el mismo interfaz de llamada y respuesta.			
Identificador	RC14	**Prioridad**	Alta

Tabla 15: RC14 - Sistema de gestión homogéneo y centralizado

3.2.2 Restricciones generales

A continuación se incluyen las restricciones generales identificadas durante el análisis.

Reservas vencidas			
Ninguna acción estará disponible sobre acciones cuya franja horaria haya vencido.			
Identificador	RG01	**Prioridad**	Alta

Tabla 16: RG01 – Reservas vencidas

Reservas no modificables			
Una reserva sólo podrá ser modificada hasta antes de que su franja horaria comience.			
Identificador	RG02	**Prioridad**	Alta

Tabla 17 - RG02: Reservas no modificables

Autenticación y securización			
El API tendrá sistemas de control de acceso para evitar un uso no autorizado.			
Identificador	RG03	**Prioridad**	Alta

Tabla 18: RG03 - Acceso al API

Sistema sin estado			
El API no mantendrá estados, es decir, cada llamada será considerada como independiente y contendrá todos los datos necesarios para resolver la solicitud.			
Identificador	RG04	**Prioridad**	Alta

Tabla 19: RG04 - Sistema sin estado

3.2.3 Características de los usuarios

La solución va a tener dos tipos de usuarios: aplicaciones y administradores. A continuación se especifica a qué corresponde cada uno.

3.2.3.1 Aplicaciones

La solución está destinada principalmente a un tipo de usuario. A este usuario lo vamos a llamar "Aplicación consumidora". La aplicación consumidora no va a ser un usuario final, sino que será otra sistema que va a atacar a los diferentes métodos del API. Simultáneamente puede haber una o más aplicaciones consumidoras atacando el API y estas pueden ser homogéneas o heterogéneas.

3.2.3.2 Administradores

El segundo tipo de usuarios que interactuará con la solución son los "Administradores". Estos usuarios realizan un uso puntual de determinadas funcionalidades de la aplicación destinado a su mantenimiento.

3.4 Casos de uso

Los casos de uso recogen los diferentes escenarios en los que tendrán lugar los requisitos de usuarios.

A continuación se incluye el diagrama con todos los casos de uso del sistema y los actores que los utilizarán:

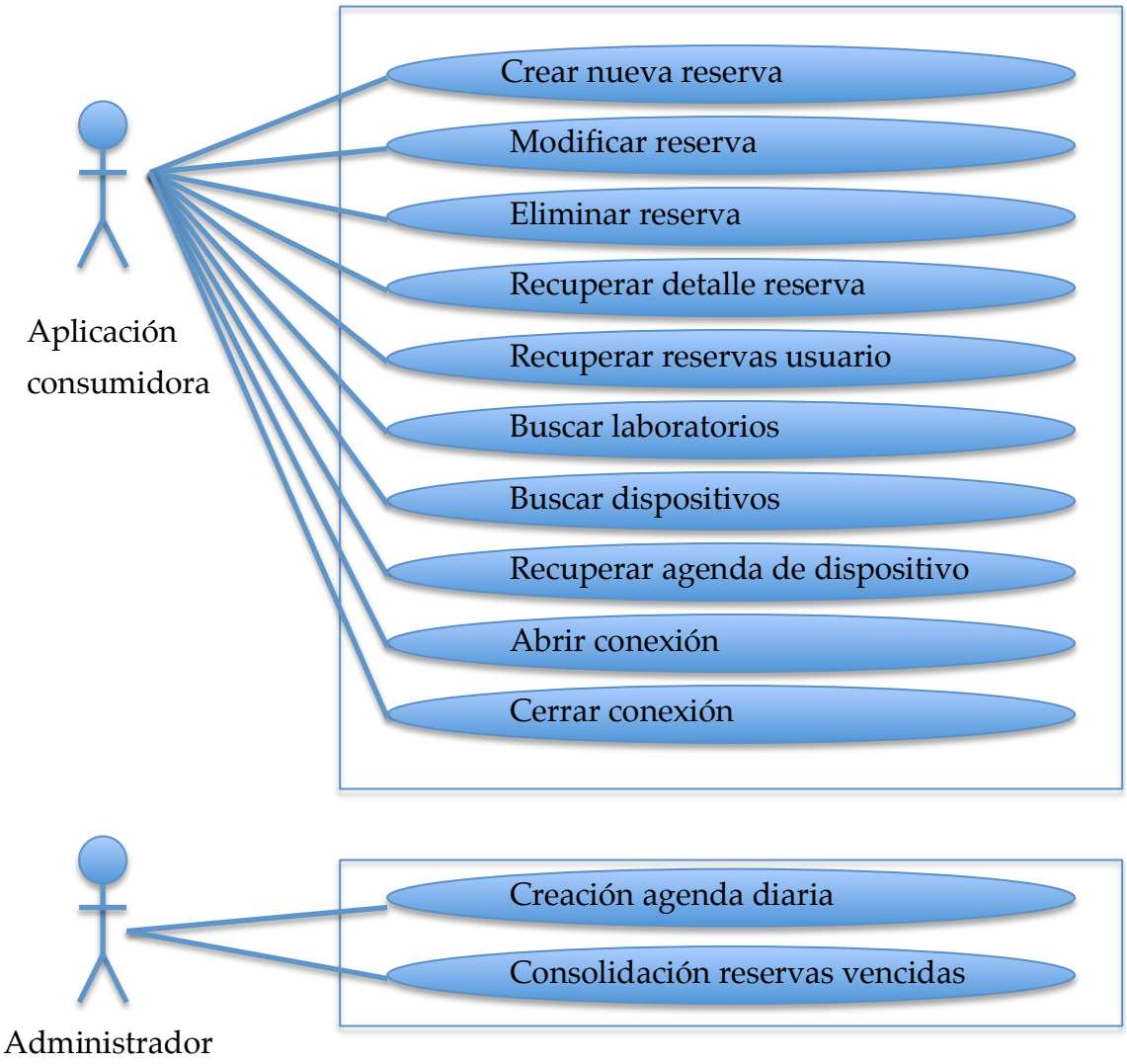


Figura 7: Diagrama de casos de uso

A continuación se desglosa cada uno de los casos de uso con toda su información relevante.

CU01 - Crear nueva reserva	
El usuario solicita crear nueva reserva.	
Actores	Aplicación consumidora
Precondiciones	El sistema llamante debe tener las credenciales necesarias para considerarse apto para realizar la petición. La llamada al método debe incluir todos los datos necesarios según la especificación del API.
Postcondiciones	Si todos los datos son correctos, la reserva quedará guardada en el sistema. El dispositivo ya no estará disponible para reservas en el mismo horario.
Escenario	

• El sistema llamante hace una llamada al método Crear nueva reserva del API incluyendo los datos necesarios:
  ○ Dispositivo
  ○ Usuario
  ○ Fecha
  ○ Horario de inicio
  ○ Horario de fin
• El sistema inserta en la base de datos la nueva reserva.
• El sistema devuelve el identificador de la reserva creada en la respuesta.	

Tabla 20: CU01 - Crear nueva reserva

CU02 - Modificar reserva	
El usuario solicita modificar una reserva ya existente.	
Actores	Aplicación consumidora
Precondiciones	El sistema llamante debe tener las credenciales necesarias para considerarse apto para realizar la petición. La llamada al método debe incluir todos los datos necesarios según la especificación del API. La reserva a modificar debe existir y no haber vencido todavía.
Postcondiciones	Si todos los datos son correctos, la reserva quedará modificada en el sistema. El dispositivo volverá a estar disponible para reservas en el horario inicial y dejará de estarlo en el nuevo horario.
Escenario	

• El sistema llamante hace una llamada al método Modificar reserva del API incluyendo los datos necesarios:
  ○ Identificador de reserva
  ○ Dispositivo
  ○ Usuario
  ○ Fecha
  ○ Horario de inicio
  ○ Horario de fin
• El sistema inserta en la base de datos la nueva reserva.
• El sistema elimina de la base de datos la antigua reserva
• El sistema devuelve un OK como respuesta.	

Tabla 21: CU02 - Modificar reserva

CU03 - Eliminar reserva	
El usuario solicita eliminar una reserva ya existente.	
Actores	Aplicación consumidora
Precondiciones	El sistema llamante debe tener las credenciales necesarias para considerarse apto para realizar la petición. La llamada al método debe incluir todos los datos necesarios según la especificación del API. La reserva a eliminar debe existir y no haber vencido todavía.
Postcondiciones	Si todos los datos son correctos, la reserva quedará eliminada del sistema. El dispositivo volverá a estar disponible para reservas en el mismo horario.
Escenario	
• El sistema llamante hace una llamada al método Eliminar reserva del API incluyendo los datos necesarios: ○ Identificador de la reserva • El sistema elimina de la base de datos la nueva reserva. • El sistema devuelve un OK como respuesta.	

Tabla 22: CU03 - Eliminar reserva

CU04 – Recuperar detalle reserva	
El usuario solicita recuperar el detalle de una reserva ya existente.	
Actores	Aplicación consumidora
Precondiciones	El sistema llamante debe tener las credenciales necesarias para considerarse apto para realizar la petición. La llamada al método debe incluir todos los datos necesarios según la especificación del API. La reserva a consultar debe existir y no haber vencido todavía.
Postcondiciones	La reserva no sufrirá modificaciones.
Escenario	

• El sistema llamante hace una llamada al método Recuperar reserva del API incluyendo los datos necesarios:
  ○ Identificador de la reserva
• El sistema consultará en la base de datos la reserva.
• El sistema devuelve los datos de la reserva como respuesta:
  ○ Identificador de la reserva
  ○ Usuario
  ○ Fecha de la reserva
  ○ Hora de comienzo
  ○ Hora de fin
  ○ Nombre del laboratorio
  ○ Descripción del laboratorio
  ○ Identificador del dispositivo
  ○ Nombre del dispositivo
  ○ Tipo dispositivo
  ○ Lista de conexiones. Por cada una de ellas:
  ▪ IP de conexión al dispositivo
  ▪ Puerto de conexión al dispositivo
  ▪ Descripción	

Tabla 23: CU04 - Recuperar detalle reserva

CU05 – Recuperar reservas de usuario	
El usuario solicita recuperar todas las reservas no vencidas de un usuario.	
Actores	Aplicación consumidora
Precondiciones	El sistema llamante debe tener las credenciales necesarias para considerarse apto para realizar la petición. La llamada al método debe incluir todos los datos necesarios según la especificación del API.
Postcondiciones	Las reservas devueltas serán únicamente las no vencidas.
Escenario	

• El sistema llamante hace una llamada al método Recuperar reserva del API incluyendo los datos necesarios:
  ○ Usuario
• El sistema consultará en la base las reservas no vencidas para ese usuario.
• El sistema devolverá una lista con todas las reservas no vencidas del usuario. Por cada reserva devolverá los siguientes datos:
  ○ Identificador de la reserva
  ○ Usuario
  ○ Fecha de la reserva
  ○ Hora de comienzo
  ○ Hora de fin
  ○ Nombre del laboratorio
  ○ Descripción del laboratorio
  ○ Identificador del dispositivo
  ○ Nombre del dispositivo
  ○ Tipo dispositivo
  ○ Lista de conexiones. Por cada una de ellas:
  ▪ IP de conexión al dispositivo
  ▪ Puerto de conexión al dispositivo
  ▪ Descripción	

Tabla 24: CU05 - Recuperar reservas de usuario

CU06 – Buscar laboratorios	
El usuario solicita recuperar todos los laboratorios disponibles..	
Actores	Aplicación consumidora
Precondiciones	El sistema llamante debe tener las credenciales necesarias para considerarse apto para realizar la petición.
Postcondiciones	Sólo se devolverán los laboratorios y dispositivos disponibles. No se realizarán modificaciones en ninguna información.
Escenario	

• El sistema llamante hace una llamada al método Buscar laboratorios del API sin parámetros.
• El sistema consultará en la base los laboratorios disponibles.
• El sistema devolverá una lista con todos los laboratorios y la lista de dispositivos disponibles en cada uno de ellos. Por cada laboratorio devolverá los siguientes datos:
  ○ Identificador del laboratorio
  ○ Nombre de laboratorio
  ○ Descripción de laboratorio
  ○ Localización del laboratorio
  ○ Lista de dispositivos. Por cada uno:
  ▪ Tipo
  ▪ Identificador del dispositivo
  ▪ Nombre del dispositivo
  ▪ Lista de conexiones. Por cada una de ellas:
  • IP de conexión al dispositivo
  • Puerto de conexión al dispositivo
  • Descripción	

Tabla 25: CU06 - Buscar laboratorios

CU07 – Buscar dispositivos	
El usuario solicita recuperar todos los dispositivos disponibles.	
Actores	Aplicación consumidora
Precondiciones	El sistema llamante debe tener las credenciales necesarias para considerarse apto para realizar la petición. La llamada al método debe incluir todos los datos necesarios según la especificación del API.
Postcondiciones	Dependiendo de los parámetros de entrada se devolverán unos u otros datos: • Identificador de laboratorio → Todos los dispositivos disponibles del laboratorio • Tipo de dispositivo → Todos los dispositivos disponibles de ese tipo. • Identificador de laboratorio y Tipo de dispositivo→ Todos los dispositivos de ese tipo en ese laboratorio • Identificador de dispositivo (con o sin otros parámetros) → Los datos del dispositivo. • Ningún parámetro → Todos los dispositivos disponibles.
Escenario	

• El sistema llamante hace una llamada al método Buscar dispositivos del API sin parámetros o incluyendo los datos necesarios para filtrar:
  ○ Identificador de laboratorio (opcional)
  ○ Tipo de dispositivo (opcional)
  ○ Identificador de dispositivo (opcional)
• El sistema consultará en la base los dispositivos disponibles coincidentes con los parámetros recibidos.
• El sistema devolverá una lista con todos los dispositivos disponibles que concuerden. Por cada dispositivo devolverá los siguientes datos:
  ○ Identificador del laboratorio
  ○ Nombre de laboratorio
  ○ Descripción de laboratorio
  ○ Localización del laboratorio
  ○ Lista de dispositivos. Por cada uno:
  ▪ Tipo
  ▪ Identificador del dispositivo	

▪ Nombre del dispositivo ▪ Lista de conexiones. Por cada una de ellas: • IP de conexión al dispositivo • Puerto de conexión al dispositivo • Descripción

Tabla 26: CU07 - Buscar dispositivos

CU08 – Recuperar agenda de dispositivo	
El usuario solicita recuperar los horarios disponibles para reserva de un dispositivo.	
Actores	Aplicación consumidora
Precondiciones	El sistema llamante debe tener las credenciales necesarias para considerarse apto para realizar la petición. La llamada al método debe incluir todos los datos necesarios según la especificación del API. El dispositivo sobre el que se solicita la agenda debe estar disponible.
Postcondiciones	No se realizarán modificaciones en ninguna información
Escenario	
• El sistema llamante hace una llamada al método Recuperar agenda del dispositivo del API incluyendo los datos necesarios: o Identificador de dispositivo • El sistema consultará en la base los horarios disponibles en la agenda del dispositivo. • El sistema devolverá una lista con todos los horarios disponibles. Será una lista de: o Fecha o Lista de horarios disponibles ▪ Horario de inicio ▪ Horario de fin	

Tabla 27: CU08 - Recuperar agenda de dispositivo

CU09 – Abrir conexión	
El usuario solicita abrir una conexión con un dispositivo y recuperar los datos de conexión al mismo.	
Actores	Aplicación consumidora
Precondiciones	El sistema llamante debe tener las credenciales necesarias para considerarse apto para realizar la petición. La llamada al método debe incluir todos los datos necesarios según la especificación del API. La reserva sobre la que se solicita la acción debe estar dentro de la franja horaria de la reserva.
Postcondiciones	El sistema remoto habrá sido informado de que debe empezar a emitir en streaming y abrir un socket y quedarse a la espera.
Escenario	

- El sistema llamante hace una llamada al método Abrir conexión del API incluyendo los datos necesarios:
  - Identificador de reserva
- El sistema consultará en la base detalles de conexión con el dispositivo correspondiente a la reserva.
- El sistema enviará mensaje al dispositivo para que empiece a emitir en streaming y abra un socket en el puerto especificado y se quede a la espera.
- El sistema guardará los datos de la conexión en la base de datos.
- El sistema devolverá los datos de conexión:
  - Identificador de la reserva
  - Usuario
  - Fecha de la reserva
  - Hora de comienzo
  - Hora de fin
  - Identificador del dispositivo
  - Nombre del dispositivo
  - Tipo dispositivo
  - Lista de conexiones. Por cada una de ellas:
    - IP de conexión al dispositivo
    - Puerto de conexión al dispositivo
    - Descripción

Tabla 28: CU09 - Abrir conexión

CU10 – Cerrar conexión	
El usuario solicita cerrar la conexión con el dispositivo.	
Actores	Aplicación consumidora
Precondiciones	El sistema llamante debe tener las credenciales necesarias para considerarse apto para realizar la petición. La llamada al método debe incluir todos los datos necesarios según la especificación del API. La reserva sobre la que se solicita la acción debe estar dentro de la franja horaria de la reserva.
Postcondiciones	El sistema remoto habrá sido informado de que debe cerrar la conexión socket y dejar de emitir en streaming.
Escenario	
• El sistema llamante hace una llamada al método Cerrar conexión del API incluyendo los datos necesarios: ○ Identificador de la conexión. • El sistema consultará en la base detalles de conexión con el dispositivo correspondiente a la conexión. • El sistema enviará mensaje al dispositivo para que cierre todas las conexiones: socket de datos y streaming de video. • El sistema eliminará la conexión de la base de datos. • El sistema devuelve un OK como respuesta.	

Tabla 29: CU10 - Cerrar conexión

CU11 – Creación agencia diaria	
El usuario genera la nueva agenda del periodo dado para el dispositivo.	
Actores	Administradores
Precondiciones	
Postcondiciones	Los horarios disponibles para la siguiente semana quedarán preparados.
Escenario	
• El administrador lanzará el proceso de generación de agenda indicando el parámetro de periodo. • El proceso revisará toda la agenda de todos dispositivos y creará en la base de datos aquellas horas o días no generados hasta completar el periodo indicado.	

Tabla 30: CU11 - Creación agenda diaria

CU12 – Consolidación reservas vencidas	
El usuario solicita archivar las reservas ya pasadas.	
Actores	Administradores
Precondiciones	
Postcondiciones	La agenda de reservas quedará únicamente con los huecos disponibles y las reservas no vencidas.
Escenario	
• El administrador lanzará el proceso de consolidación de reservas vencidas. • El proceso revisará toda la agenda de todos dispositivos y copiará las reservas ya vencidas a la tabla histórico borrándolas de la tabla de reservas.	

Tabla 31: CU12 - Consolidación reservas vencidas

3.5 Requisitos del software

Los requisitos de software trasladan los requisitos de usuario y los escenarios de caso de uso, que están expresados en un lenguaje de negocio, a una definición de alto nivel funcional que permita ser la base para la definición de la solución.

Hay dos tipos de requisitos de software principalmente. El primero son los requisitos funcionales, que expresan cada uno de los comportamientos que deberán ser implementados en la solución. El segundo son los requisitos no funcionales que definen cómo deberán ser los interfaces de la solución y su implementación.

3.5.1 Requisitos funcionales

Acción crear nueva reserva			
El sistema guardará en base de datos los datos la nueva reserva. Si la inserción se ha producido con éxito devolverá un OK. Si la inserción no se ha producido con éxito devolverá un error.			
Identificador	RF01	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC01, RC13, RG01, RG04

Tabla 32: RF01 - Acción crear nueva reserva

Acción modificar reserva			
El sistema modificará la reserva pasada por parámetros si ya existe en la base de datos con los datos recibidos como parámetros. La modificación podrá afectar a 1 o más valores de la reserva. Los recursos de la nueva reserva serán bloqueados y los recursos de la reserva original serán liberados. Si la reserva no existiera en el sistema no se creará una nueva y se considerará un error. Si la modificación se ha producido con éxito devolverá un OK. Si la modificación no se ha producido con éxito devolverá un error.			
Identificador	RF02	**Necesidad**	Opcional
Prioridad	Media	**Origen**	RC02, RC13, RG01, RG02, RG04

Tabla 33: RF02 - Acción modificar reserva

Acción eliminar reserva			
El sistema eliminará la reserva pasada por parámetros si ya existe en la base de datos y no ha vencido. Si la eliminación se ha producido con éxito devolverá un OK. Si la eliminación no se ha producido con éxito devolverá un error.			
Identificador	RF03	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC03, RC13, RG01, RG04

Tabla 34: RF03 - Acción eliminar reserva

Acción recuperar detalle reserva			
El sistema recuperará de base de datos la reserva pasada por parámetros si ya existe en la base de datos y no ha vencido. Si la reserva no existiera se considerará un error. Si la consulta ha devuelto resultados se extraerá la siguiente información: • Identificador de la reserva → Dato alfanumérico • Identificador de usuario → Dato alfanumérico con formato email • Fecha → Fecha (sin hora) • Horario de inicio → Hora • Horario de fin → Hora • Nombre del laboratorio → Cadena de caracteres • Descripción del laboratorio → Cadena de caracteres • Identificador del dispositivo → Dato alfanumérico • Nombre del dispositivo → Cadena de caracteres • Tipo dispositivo → Cadena de caracteres • Lista de conexiones. Por cada una de ellas: o IP de conexión al dispositivo → Dato alfanumérico con formato de dirección IP o Puerto de conexión al dispositivo → Dato numérico o Descripción → Cadena de caracteres Se formatearán dichos datos y se devolverán como respuesta. Si la consulta no ha producido resultados devolverá un error.			
Identificador	RF04	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC04, RC13, RG01, RG04

Tabla 35: RF04 - Acción recuperar detalle reserva

Acción recuperar reservas usuario			
El sistema recuperará de base de datos todas las reserva no vencidas del usuario pasado por parámetros. Si el usuario no existiera se considerará un error. Si la consulta ha devuelto resultados se extraerá una lista con todas las reservas no vencidas del usuario con la siguiente información por cada reserva: • Identificador de la reserva → Dato alfanumérico • Identificador de usuario → Dato alfanumérico con formato email • Fecha → Fecha (sin hora) • Horario de inicio → Hora • Horario de fin → Hora • Nombre del laboratorio → Cadena de caracteres • Descripción del laboratorio → Cadena de caracteres • Identificador del dispositivo → Dato alfanumérico • Nombre del dispositivo → Cadena de caracteres • Tipo dispositivo → Cadena de caracteres • Lista de conexiones. Por cada una de ellas: o IP de conexión al dispositivo → Dato alfanumérico con formato de dirección IP o Puerto de conexión al dispositivo → Dato numérico o Descripción → Cadena de caracteres Se formatearán dichos datos y se devolverán como respuesta. Si el usuario existe pero no tiene reservas se considerará un éxito y se devolverá una lista vacía. Si la consulta no se puede resolver devolverá un error.			
Identificador	RF05	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC05, RC13, RG01, RG04

Tabla 36: RF05 - Acción recuperar reservas usuario

Acción buscar laboratorios			

El sistema realizará una consulta a base de datos para obtener todos los laboratorios disponibles (que tengan a true el campo "available").
Por cada laboratorio extraerá la siguiente información:
• Identificador del laboratorio → Dato alfanumérico
• Nombre del laboratorio → Cadena de caracteres
• Descripción del laboratorio → Cadena de caracteres
• Localización del laboratorio → Cadena de caracteres
• Lista de dispositivos. Por cada uno:
  ○ Identificador del dispositivo → Dato alfanumérico
  ○ Nombre del dispositivo → Cadena de caracteres
  ○ Tipo dispositivo → Cadena de caracteres
  ○ Lista de conexiones → Colección. Por cada una de ellas:
  ▪ IP de conexión al dispositivo → Dato alfanumérico con formato de dirección IP
  ▪ Puerto de conexión al dispositivo → Dato numérico
  ▪ Descripción → Cadena de caracteres
Se formatearán dichos datos y se devolverán como respuesta.
Si no hubiera laboratorios disponibles se devolverá una lista vacía pero no se considerará un error.			

Identificador	RF06	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC06, RC13, RG04

Tabla 37: RF06 - Acción buscar laboratorios

Acción buscar dispositivos			
El sistema realizará una consulta a base de datos para obtener todos los dispositivos de acuerdo a las siguientes reglas: • Identificador de laboratorio (Dato alfanumérico) → Se buscarán todos los dispositivos disponibles del laboratorio pasado como parámetro • Tipo de dispositivo (Cadena de caracteres) → Se buscarán todos los dispositivos disponibles del tipo pasado como parámetro • Identificador de laboratorio (Dato alfanumérico) y Tipo de dispositivo (Cadena de caracteres)→ Se buscarán todos los dispositivos del tipo pasado como parámetro en el laboratorio pasado como parámetro. • Identificador de dispositivo (con o sin otros parámetros) (Dato alfanumérico) → Se buscarán todos los datos para el dispositivo concreto pasado como parámetro • Ningún parámetro → Todos los dispositivos disponibles. Se extraerá una lista de dispositivos de la base de datos con la siguiente información por cada uno de ellos: • Identificador del laboratorio → Dato alfanumérico • Nombre del laboratorio → Cadena de caracteres • Descripción del laboratorio → Cadena de caracteres • Localización del laboratorio → Cadena de caracteres • Lista de dispositivos. Por cada uno: o Identificador del dispositivo → Dato alfanumérico o Nombre del dispositivo → Cadena de caracteres o Tipo dispositivo → Cadena de caracteres o Lista de conexiones → Colección. Por cada una de ellas: ▪ IP de conexión al dispositivo → Dato alfanumérico con formato de dirección IP ▪ Puerto de conexión al dispositivo → Dato numérico ▪ Descripción → Cadena de caracteres Se formatearán dichos datos y se devolverán como respuesta. Si no hubiera dispositivos disponibles con los parámetros de Tipo y/o Laboratorio se devolverá una lista vacía pero no se considerará un error. Si se indica un dispositivo concreto y no existe, se devolverá un error.			
Identificador	RF07	**Necesidad**	Opcional
Prioridad	Baja	**Origen**	RC07, RC13, RG04

Tabla 38: RF07 - Acción buscar dispositivos

Acción recuperar agenda de dispositivo			

El sistema buscará en la base de datos todos los horarios disponibles en la agenda del dispositivo pasado como parámetro.
Se extraerá de base de datos una lista con todos los horarios disponibles compuesta de:
• Fecha → Fecha (sin hora)
• Lista de horarios disponibles → Colección:
  o Horario de inicio → Hora
  o Horario de fin → Hora
Se formatearán dichos datos y se devolverán como respuesta.
Si el dispositivo no existiera se considerará un error.			

Identificador	RF08	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC08, RC13, RG04

Tabla 39: RF08 - Acción recuperar agenda de dispositivo

Acción abrir conexión			
El sistema consultará en base de datos los datos de configuración del dispositivo correspondiente a la reserva pasada como parámetro . Con dichos datos el sistema enviará una lista de órdenes al dispositivo de laboratorio compuesta por estos tipos: - Abrir socket de comunicación de datos en IP y puerto indicados - Abrir streaming de video y comenzar a emitir en la IP y puerto indicados. El sistema devolverá como respuesta el identificador de la conexión: • Identificador de la conexión→ Dato alfanumérico Si el dispositivo o la reserva no existieran o la reserva no estuviera dentro de su franja de ejecución se devolverá un error			
Identificador	RF09	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC09, RC13, RG01, RG04

Tabla 40: RF09 - Acción abrir conexión

Acción cerrar conexión			
El sistema consultará en base de datos los datos de configuración del dispositivo correspondiente a identificador de conexión como parámetro Con dichos datos el sistema enviará una lista de órdenes al dispositivo de laboratorio compuesta por estos tipos: - Cerrar socket de comunicación de datos en IP y puerto indicados - Cerrar streaming de video y dejar de emitir en la IP y puerto indicados. Si la ejecución finaliza correctamente se devolverá un OK. Si la ejecución no finaliza correctamente se devolverá un error.			
Identificador	RF10	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC10, RC13, RG01, RG04

Tabla 41: RF10 - Acción cerrar conexión

Acción creación agenda diaria			
El sistema recorrerá la agenda disponible para el dispositivo desde la fecha y hora actual hasta la fecha y hora objetivo indicada como parámetro y creará todos aquellas nuevas entradas necesarias para completarla. Cada nueva entrada se compondrá de: • Fecha → Fecha (sin hora) • Horario de inicio → Hora • Horario de fin → Hora Se considerarán entradas ya creadas tanto aquellas vacías como aquellas que ya tengan reservas. Se considerarán entradas a crear aquellas que no figuren en la tabla de agenda de ninguna manera. Si el proceso finaliza correctamente se enviará como estado final un OK. Si el proceso no finaliza correctamente se enviará como estado final un KO. Si no es necesario crear ninguna entrada porque todas estuvieran ya creadas se entenderá que es una ejecución satisfactoria y se indicará como estado final un OK.			
Identificador	RF11	**Necesidad**	Deseable
Prioridad	Media	**Origen**	RC11, RG04

Tabla 42: RF11 - Acción creación agenda diaria

Acción consolidación reservas vencidas			
El sistema copiará las entradas ya vencidas con reserva de la agenda del dispositivo de la tabla de agenda a la tabla de histórico. La información en la tabla origen y la tabla destino será la misma: • Identificador de la reserva → Dato alfanumérico • Identificador de usuario → Dato alfanumérico con formato email • Fecha → Fecha (sin hora) • Horario de inicio → Hora • Horario de fin → Hora • Nombre del laboratorio → Cadena de caracteres • Descripción del laboratorio → Cadena de caracteres • Identificador del dispositivo → Dato alfanumérico • Nombre del dispositivo → Cadena de caracteres • Tipo dispositivo → Cadena de caracteres • Lista de conexiones. Por cada una de ellas: o IP de conexión al dispositivo → Dato alfanumérico con formato de dirección IP o Puerto de conexión al dispositivo → Dato numérico o Descripción → Cadena de caracteres La entradas sin reserva no serán copiadas a la tabla histórico. Una vez finalizada la copia, se eliminarán todas las entradas vencidas (con reserva o libres) de la agenda del dispositivo. Se entiende por entrada vencida aquella que tiene fecha y hora anterior a la actual. Si el proceso finaliza correctamente se enviará como estado final un OK. Si el proceso no finaliza correctamente se enviará como estado final un KO. Si no es necesario mover ninguna entrada porque todas estuvieran en vigor se entenderá que es una ejecución satisfactoria y se indicará como estado final un OK.			
Identificador	RF12	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC12, RG04

Tabla 43: RF12 - Acción consolidación reservas vencidas

3.5.2 Requisitos no funcionales

Interfaz método crear nueva reserva			
El API deberá implementar un método denominado "Crear nueva reserva" que reciba como parámetros: o Identificador de dispositivo → Dato alfanumérico o Identificador de usuario → Dato alfanumérico con formato email o Fecha → Fecha (sin hora) o Horario de inicio → Hora o Horario de fin → Hora Cuando este método sea invocado se ejecutará la acción "Crear nueva reserva". Este método devolverá como respuesta el identificador de la reserva en caso de terminar bien o un error en caso contrario.			
Identificador	RNF01	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC01, RC13, RC14, RG03

Tabla 44: RNF01 – Interfaz método crear nueva reserva

Interfaz método modificar reserva			
El API deberá implementar un método denominado "Modificar reserva" que reciba como parámetros: o Identificador de la reserva → Dato alfanumérico o Identificador de dispositivo → Dato alfanumérico o Identificador de usuario → Dato alfanumérico con formato email o Fecha → Fecha (sin hora) o Horario de inicio → Hora o Horario de fin → Hora Cuando este método sea invocado se ejecutará la acción "Modificar reserva". Este método devolverá como respuesta un OK en caso de terminar bien o un error en caso contrario.			
Identificador	RNF02	**Necesidad**	Opcional
Prioridad	Media	**Origen**	RC02, RC13, RC14, RG03

Tabla 45: RNF02 - Interfaz método modificar reserva

Interfaz método eliminar reserva			
El API deberá implementar un método denominado "Eliminar reserva" que reciba como parámetro el identificador de la reserva. Cuando este método sea invocado se ejecutará la acción "Eliminar reserva". Este método devolverá como respuesta un OK en caso de terminar bien o un error en caso contrario.			
Identificador	RNF03	**Necesidad**	Deseable
Prioridad	Alta	**Origen**	RC03, RC13, RC14, RG03

Tabla 46: RNF03 - Interfaz método eliminar reserva

Interfaz método recuperar detalle reserva			
El API deberá implementar un método denominado "Recuperar detalle reserva" que reciba como parámetros el identificador de la reserva. Cuando este método sea invocado se ejecutará la acción "Recuperar detalle reserva". Este método devolverá como respuesta los datos de la reserva en caso de terminar bien: • Identificador de la reserva → Dato alfanumérico • Identificador de usuario → Dato alfanumérico con formato email • Fecha → Fecha (sin hora) • Horario de inicio → Hora • Horario de fin → Hora • Nombre del laboratorio → Cadena de caracteres • Descripción del laboratorio → Cadena de caracteres • Identificador del dispositivo → Dato alfanumérico • Nombre del dispositivo → Cadena de caracteres • Tipo dispositivo → Cadena de caracteres • Lista de conexiones. Por cada una de ellas: o IP de conexión al dispositivo → Dato alfanumérico con formato de dirección IP o Puerto de conexión al dispositivo → Dato numérico o Descripción → Cadena de caracteres En caso contrario devolverá un error.			
Identificador	RNF04	**Necesidad**	Deseable
Prioridad	Media	**Origen**	RC04, RC13, RC14, RG03

Tabla 47: RNF04 - Interfaz método recuperar detalle reserva

Interfaz método recuperar reservas de usuario			
El API deberá implementar un método denominado "Recuperar reservas de usuario" que reciba como parámetros: o Identificador de usuario → Dato alfanumérico con formato email Cuando este método sea invocado se ejecutará la acción "Recuperar reservas de usuario". reserva". Este método devolverá como respuesta devolverá una lista con todas las reservas no vencidas del usuario en caso de terminar bien con estos datos por cada reserva: • Identificador de la reserva → Dato alfanumérico • Identificador de usuario → Dato alfanumérico con formato email • Fecha → Fecha (sin hora) • Horario de inicio → Hora • Horario de fin → Hora • Nombre del laboratorio → Cadena de caracteres • Descripción del laboratorio → Cadena de caracteres • Identificador del dispositivo → Dato alfanumérico • Nombre del dispositivo → Cadena de caracteres • Tipo dispositivo → Cadena de caracteres • Lista de conexiones. Por cada una de ellas: o IP de conexión al dispositivo → Dato alfanumérico con formato de dirección IP o Puerto de conexión al dispositivo → Dato numérico o Descripción → Cadena de caracteres En caso contrario devolverá un error.			
Identificador	RNF05	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC05, RC13, RC14, RG03

Tabla 48: RNF05 - Interfaz método recuperar reservas de usuario

Interfaz método buscar laboratorios			
El API deberá implementar un método denominado "Buscar laboratorios" sin parámetros. Cuando este método sea invocado se ejecutará la acción "Buscar laboratorios". Este método devolverá como respuesta una lista con todos los laboratorios y la lista de dispositivos disponibles en cada uno de ellos en caso de terminar bien. Por cada laboratorio devolverá los siguientes datos: • Identificador del laboratorio → Dato alfanumérico • Nombre del laboratorio → Cadena de caracteres • Descripción del laboratorio → Cadena de caracteres • Localización del laboratorio → Cadena de caracteres • Lista de dispositivos. Por cada uno: o Identificador del dispositivo → Dato alfanumérico o Nombre del dispositivo → Cadena de caracteres o Tipo dispositivo → Cadena de caracteres o Lista de conexiones → Colección. Por cada una de ellas: ▪ IP de conexión al dispositivo → Dato alfanumérico con formato de dirección IP ▪ Puerto de conexión al dispositivo → Dato numérico ▪ Descripción → Cadena de caracteres En caso contrario devolverá un error.			
Identificador	RNF06	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC06, RC13, RC14, RG03

Tabla 49: RNF06 - Interfaz método buscar laboratorios

Interfaz método buscar dispositivos			

El API deberá implementar un método denominado "Buscar dispositivos" que puede recibir la siguiente combinación de parámetros:
• Identificador de laboratorio (Dato alfanumérico) → Todos los dispositivos disponibles del laboratorio
• Tipo de dispositivo (Cadena de caracteres) → Todos los dispositivos disponibles de ese tipo.
• Identificador de laboratorio (Dato alfanumérico) y Tipo de dispositivo (Cadena de caracteres)→ Todos los dispositivos de ese tipo en ese laboratorio
• Identificador de dispositivo (con o sin otros parámetros) (Dato alfanumérico) → Los datos del dispositivo.
• Ningún parámetro → Todos los dispositivos disponibles.
 	
Cuando este método sea invocado se ejecutará la acción "Buscar dispositivos". Este método devolverá como respuesta una lista con todos los dispositivos disponibles que concuerden con los parámetros en caso de terminar bien. Por cada dispositivo devolverá los siguientes datos:
• Identificador del laboratorio → Dato alfanumérico
• Nombre del laboratorio → Cadena de caracteres
• Descripción del laboratorio → Cadena de caracteres
• Localización del laboratorio → Cadena de caracteres
• Lista de dispositivos. Por cada uno:
  ○ Identificador del dispositivo → Dato alfanumérico
  ○ Nombre del dispositivo → Cadena de caracteres
  ○ Tipo dispositivo → Cadena de caracteres
  ○ Lista de conexiones → Colección. Por cada una de ellas:
  ▪ IP de conexión al dispositivo → Dato alfanumérico con formato de dirección IP
  ▪ Puerto de conexión al dispositivo → Dato numérico
  ▪ Descripción → Cadena de caracteres
En caso contrario devolverá un error.			

Identificador	RNF07	**Necesidad**	Opcional
Prioridad	Baja	**Origen**	RC07, RC13, RC14, RG03

Tabla 50: RNF07 - Interfaz método buscar dispositivos

Método Interfaz recuperar agenda de dispositivo			

El API deberá implementar un método denominado "Recuperar agenda de dispositivo" que reciba como parámetros:
• Identificador del dispositivo → Dato alfanumérico
Cuando este método sea invocado se ejecutará la acción "Recuperar agenda de dispositivo".
Este método devolverá como respuesta una lista con todos los horarios disponibles en caso de terminar bien. Será una lista compuesta de:
• Fecha → Fecha (sin hora)
• Lista de horarios disponibles → Colección:
  ○ Horario de inicio → Hora
  ○ Horario de fin → Hora
En caso contrario devolverá un error.			

Identificador	RNF08	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC08, RC13, RC14, RG03

Tabla 51: RNF08 – Interfaz método Interfaz recuperar agenda de dispositivo

Interfaz método abrir conexión			
El API deberá implementar un método denominado "Abrir conexión" que reciba como parámetros: • Identificador del dispositivo → Dato alfanumérico Cuando este método sea invocado se ejecutará la acción "Abrir conexión". Este método devolverá como respuesta en caso de terminar bien: • Identificador de la conexión→ Dato alfanumérico En caso contrario devolverá un error			
Identificador	RNF09	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC09, RC13, RC14, RG03

Tabla 52: RNF09 - Interfaz método abrir conexión

Interfaz método cerrar conexión			
El API deberá implementar un método denominado “Cerrar conexión” que reciba como parámetros: • Identificador de la conexión→ Dato alfanumérico Cuando este método sea invocado se ejecutará la acción “Cerrar conexión”. Este método devolverá como respuesta un OK en caso de terminar bien o un KO en caso contrario.			
Identificador	RNF10	**Necesidad**	Esencial
Prioridad	Alta	**Origen**	RC10, RC13, RC14, RG03

Tabla 53: RNF10 - Interfaz método cerrar conexión

3.5 Matriz de trazabilidad

La matriz de trazabilidad relaciona los requisitos de software con los requisitos de usuario que los definen.

Cada requisito de usuario debe tener, al menos, un requisito de software y viceversa. En caso contrario alguna de las necesidades no se verían resueltas en la solución.

	RF01	RF02	RF03	RF04	RF05	RF06	RF07	RF08	RF09	RF10	RF11	RF12	RNF01	RNF02	RNF03	RNF04	RNF05	RNF06	RNF07	RNF08	RNF09	RNF10
RC01	X												X									
RC02		X												X								
RC03			X												X							
RC04				X												X						
RC05					X												X					
RC06						X												X				
RC07							X												X			
RC08								X												X		
RC09									X												X	
RC10										X												X
RC11											X											
RC12												X										
RC13	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X	X
RC14													X	X	X	X	X	X	X	X	X	X
RG01	X	X	X	X	X				X	X												
RG02		X																				
RG03													X	X	X	X	X	X	X	X	X	X
RG04	X	X	X	X	X	X	X	X	X	X	X	X										

4 Diseño

En el diseño de la aplicación se tratará de explicar la arquitectura que compone tanto la aplicación en sí misma como todo el ecosistema al que daría soporte, además veremos sus componentes de forma individualizada, el esquema seleccionado para la base de datos en MongoDB y la estructura de clases que implementan la solución.

4.1 Arquitectura del sistema

La arquitectura seguida en el diseño e implementación de la solución para el backend está representada en la siguiente figura:

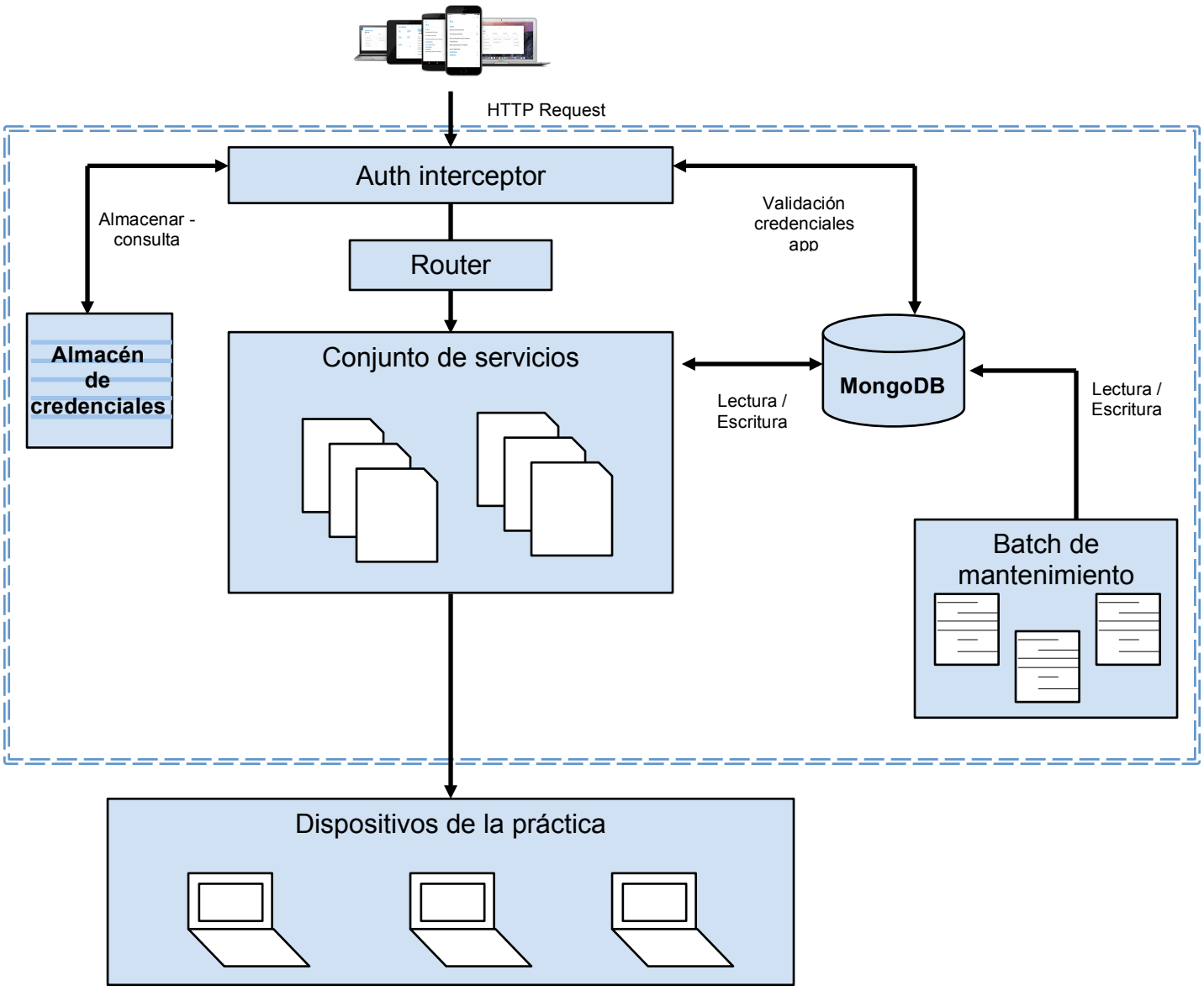


Figura 8: Arquitectura Backend

Del diagrama anterior se desprende un conjunto de componentes que van a ser explicados a continuación.

Interceptor de autenticación

Todas las llamadas que sean recibidas desde las aplicaciones remotas deberán pasar en primera instancia por esta pieza, que hará las veces de interceptor de las peticiones y validará que llevan las credenciales correctas para proceder con la petición. Para ello, y siguiendo el estándar de autenticación OAuth 1.0 que es el que se ha implementando en la solución, se validará que en la petición HTTP se disponga del "apiKey" y "apiSecret", que harán las veces del par usuario y password, y se comprobará contra la colección de usuario activos que las credenciales son correctas. En caso de dichas credenciales sean válidas se genera un token temporal que podrá ser utilizado por la aplicación llamante para futuras interacciones con el backend, para lo cual dicho token será almacenado en el backend, lo cual da la ventaja de que "apiKey" y "apiSecret" solo deberán de enviarse en la primera petición.

Almacén de credenciales

Este almacén será el que se utilice para almacenar el token autogenerado, una vez que las aplicaciones llamantes se han autenticado con el par "apiKey" y "apiSecret". Además, asociado al token se guardan los datos de la aplicación llamante para así validar que no es un tercero el que está haciendo uso del token para suplantar la identidad del llamante, tal y como se especifica en el estándar de autenticación OAuth 1.0 que es el que se ha implementado en la solución. Los datos almacenados tendrán un periodo de validez que puede ser modificado en la configuración de la aplicación y por tanto las credenciales caducarán pasado un tiempo y la aplicación llamante deberá de volver a autenticarse con su par "apiKey" y "apiSecret" generando así un nuevo token válido.

Base de datos: MongoDB

La base de datos elegida para la solución ha sido MongoDB. Debido a la facilidad y sencillez a la hora de generar los documentos y las respuestas que son devueltas a las aplicaciones llamantes. Por tanto, se almacenará toda la información que compone la gestión de reservas y la realización de prácticas en dispositivos remotos: laboratorios, dispositivos, reservas activas, agenda futura… Además, cuenta con una colección (tabla en lenguajes SQL) donde se almacena el conjunto de apiKey y apiSecret, este último cifrado, de las aplicaciones llamantes que puede tener acceso al backend, así como una pequeña descripción para cada una de ellas.

Enrutador

Una vez que las llamadas de terceras aplicaciones han sido validadas y autenticadas por el interceptor, pasan por el enrutado que contiene los patrones de las URLs llamantes y los servicios que dan respuesta a cada una de ellas. Por tanto

será esta pieza la que derive a cada servicio específico la llamada para que sea ejecutada por el backend.

Conjunto de servicios

La aplicación backend se compone de un conjunto de servicios modulares y que funcionan de manera totalmente independiente. Por tanto, sería posible añadir o eliminar servicios sin modificar el comportamiento del resto de la aplicación. Estos servicios recibirán una petición HTTP derivada desde el enrutado y le darán una respuesta normalmente consultando y/o editando la base de datos o bien comunicando con los dispositivos que se encuentran en los laboratorios para la realización remota de las prácticas.

Batch de Mantenimiento

Además de contar con el conjunto de servicios, hay que tener en cuenta que hay una serie de tareas de mantenimiento que han de realizarse con cierta frecuencia. Para ello se ha creado un lote de procesos que serán ejecutados mediante tareas "cron" de forma automatizada y que en este caso permiten la generación diaria de los huecos de la agenda para todos los dispositivos disponibles así como la consolidación de un histórico de todas las reservas pasadas efectuadas. Este conjunto de procesos podrá crecer de forma independiente en un futuro según las necesidades que se vayan encontrando en el proyecto.

Una vez que hemos desgranado las piezas que componen la aplicación de backend podemos ver todo el conjunto de la arquitectura dentro del ecosistema en el que se encuentra esta aplicación.

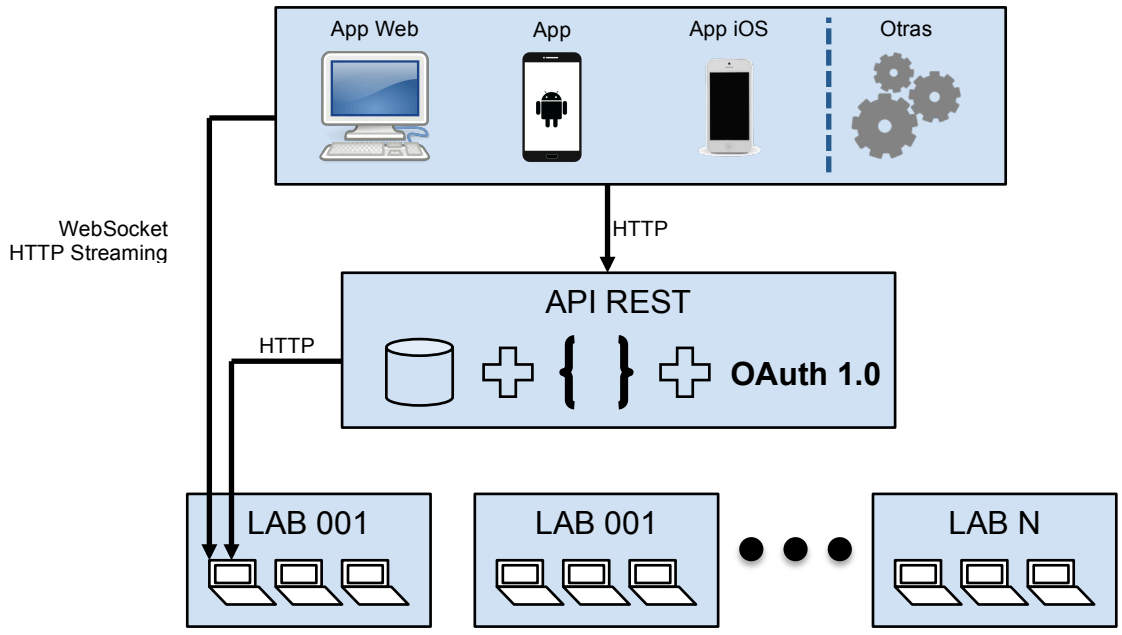


Figura 9: Arquitectura global

A grandes rasgos esta sería la visión global del proyecto, compuestas por los siguientes elementos:

Aplicaciones consumidoras

Existirá un conjunto de aplicaciones que serán las interesadas en consumir los datos del sistema centralizado de gestión, de forma que estas se independizan de todas las tareas de mantenimiento de los laboratorios, dispositivos, agenda… y se centran en la capa de presentación o integración que les interese en cada caso, además de que los datos son compartidos y sincronizados por todas ellas. En un inicio, dentro del ecosistema, sólo contamos con una aplicación web pero a futuro se podrían realizar desde aplicaciones nativas para dispositivos móviles hasta terceras aplicaciones que les interese integrar los datos proporcionados en el backend para su posterior tratamiento en otro tipo de aplicaciones.

Backend API Rest

Esta pieza será la encargada de centralizar toda la información tanto de los laboratorios/dispositivos existentes como de los reservas realizadas desde las aplicaciones de terceros. Dará un soporte a través de llamadas de tipo HTTP Rest consumiendo y produciendo siempre contenido en formato JSON, sencillo de entender y procesar al mismo tiempo. Además, cuenta con una documentación asociada que permite a los desarrolladores de esas aplicaciones llamantes conocer el conjunto de métodos que se implementan desde el API, así como los datos esperados tanto a la entrada como a la salida. Por último, realiza la labor de conexión inicial con los dispositivos, ya que desde este backend se conocen todos los datos y por tanto puede indicar cuándo arrancar/parar a los dispositivos para que reciban comunicación directa desde terceras aplicaciones.

Dispositivos

Contaremos con un conjunto de dispositivos disponibles divididos a su vez en laboratorios. Estos dispositivos serán los que reciban las instrucciones por parte de las aplicaciones desde donde se realizan las prácticas remotas, y para ello deberán de contar con uno o varios puntos de entrada al mismo. En la actualidad contamos con una IP accesible desde el exterior y un conjunto de puertos que darán servicio a una comunicación a través del protocolo Websocket y al envío de video en streaming bajo HTTP, lo cual no quiere decir que no puede ser ampliado en el futuro. Además, cada dispositivo cuenta con una interfaz API Rest mediante la que recibirá los comandos de parada y arranque que le lleguen desde la aplicación backend.

4.2 Diagrama de componentes

Este diagrama pretende representar los diferentes componentes del sistema y su relación entre ellos. En este caso concreto, al tratarse de un API y sin sesión, todos los componentes son independientes y se podrían ejecutar sin relación entre ellos. No obstante, se ha reflejado cómo sería un flujo habitual a seguir por la Aplicación consumidora.

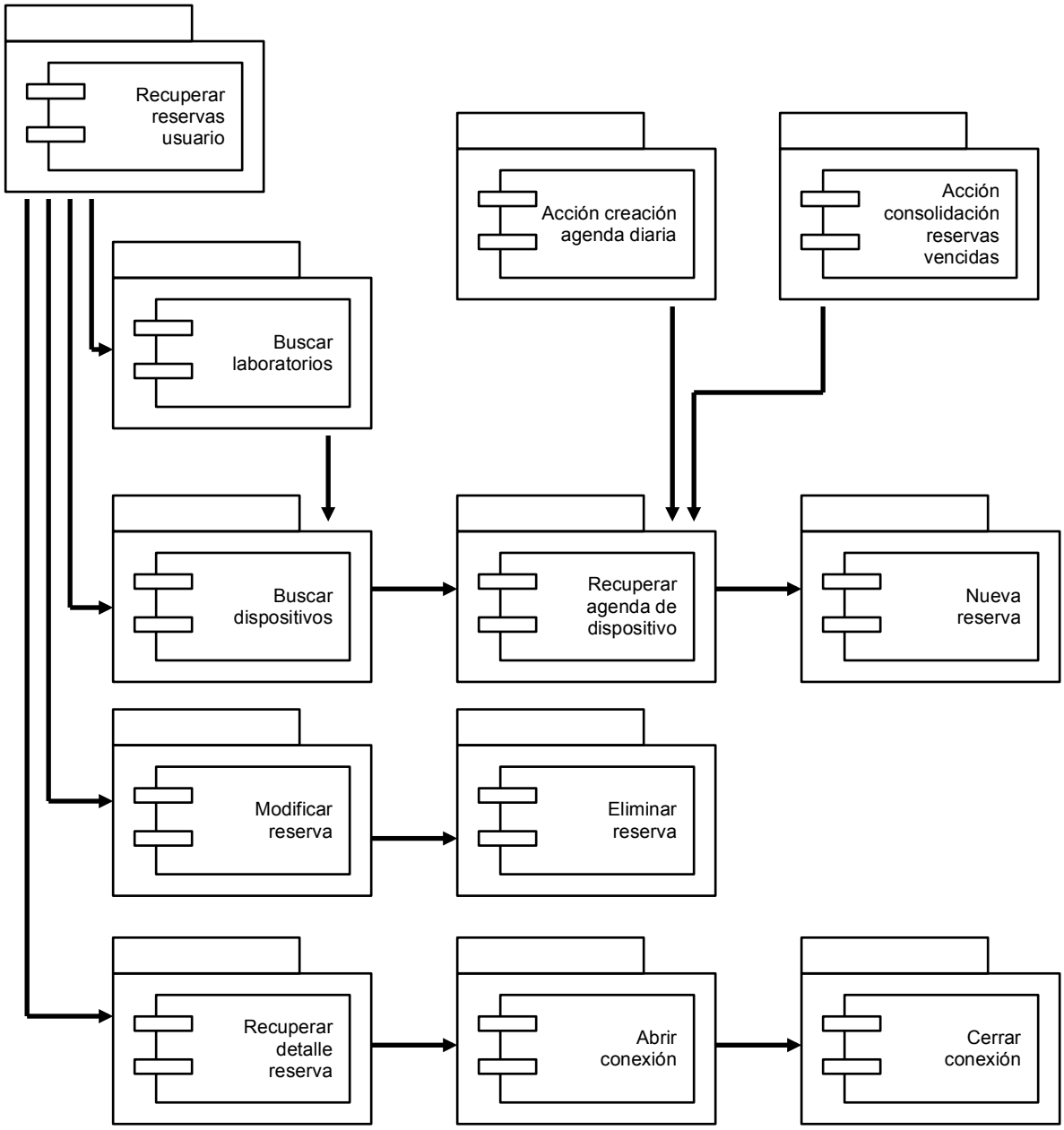


Figura 10: Diagrama de componentes

Nueva reserva	
Identificador	C01
Origen	RC01, RC13, RC14, RG01, RG03, RG04
Objetivo	Crear una nueva reserva.
Acciones	
- Validar los parámetros recibidos. - Almacenar la nueva reserva. - Componer la respuesta y devolverla.	

Tabla 54: C01 – Nueva reserva

Modificar reserva	
Identificador	C02
Origen	RC02, RC13, RC14, RG01, RG02, RG03, RG04
Objetivo	Modificar una reserva ya existente.
Acciones	
- Validar los parámetros recibidos. - Modificar la reserva ya existente. - Componer la respuesta y devolverla.	

Tabla 55: C02 – Modificar reserva

Eliminar reserva	
Identificador	C03
Origen	RC03, RC13, RC14, RG01, RG03, RG04
Objetivo	Eliminar una reserva ya existente.
Acciones	
- Validar los parámetros recibidos. - Eliminar la reserva ya existente. - Componer la respuesta y devolverla.	

Tabla 56: C03 – Eliminar reserva

Recuperar detalle reserva	
Identificador	C04
Origen	RC04, RC13, RC14, RG01, RG03, RG04
Objetivo	Recuperar toda la información relativa a una reserva.
Acciones	
- Validar los parámetros recibidos. - Recuperar la información de la reserva dada. - Componer la respuesta y devolverla.	

Tabla 57: C04 – Recuperar detalle reserva

Recuperar reservas usuario	
Identificador	C05
Origen	RC05, RC13, RC14, RG01, RG03, RG04
Objetivo	Recuperar la información de todas las reservas de un usuario.
Acciones	
- Validar los parámetros recibidos. - Recuperar la información de todas las reservas del usuario dado. - Componer la respuesta y devolverla.	

Tabla 58: C05 – Recuperar reservas usuario

Buscar laboratorios	
Identificador	C06
Origen	RC06, RC13, RC14, RG03, RG04
Objetivo	Recuperar los laboratorios disponibles.
Acciones	
- Validar los parámetros recibidos. - Recuperar la información de todos los laboratorios disponibles. - Componer la respuesta y devolverla.	

Tabla 59: C06 – Buscar laboratorios

Buscar dispositivos	
Identificador	C07
Origen	RC07, RC13, RC14, RG03, RG04
Objetivo	Buscar los dispositivos disponibles que concuerden con los parámetros.
Acciones	
- Validar los parámetros recibidos. - Si no se reciben parámetros: recuperar la lista de todos los dispositivos disponibles. - Si se indica laboratorio: recuperar la lista de todos los dispositivos disponibles del laboratorio indicado con su información. - Si se indica tipo de dispositivo: recuperar la lista de todos los dispositivos disponibles de ese tipo con su información. - Si se indica laboratorio y tipo de dispositivo: recuperar la lista de todos los dispositivos disponibles de ese tipo en ese laboratorio con su información. - Si se indica el identificador de dispositivo (independientemente de si hay más parámetros: recuperar la información del dispositivo dado. - Componer la respuesta y devolverla.	

Tabla 60: C07 – Buscar dispositivos

Recuperar agenda dispositivo	
Identificador	C08
Origen	RC08, RC13, RC14, RG03, RG04
Objetivo	Recuperar los horarios disponibles del dispositivo dado.
Acciones	
- Validar los parámetros recibidos. - Recuperar todos los horarios disponibles para el dispositivo dado. - Componer la respuesta y devolverla.	

Tabla 61: C08 – Recuperar agenda dispositivo

Abrir conexión	
Identificador	C09
Origen	RC09, RC13, RC14, RG01, RG03, RG04
Objetivo	Recuperar los datos de conexión e indicar al dispositivo de laboratorio que inicie las conexiones.
Acciones	
- Validar los parámetros recibidos. - Recuperar los datos de conexión para la reserva especificada. - Enviar mensaje al dispositivo para que inicie streaming y socket. - Almacenar la información de la conexión. - Componer la respuesta y devolverla.	

Tabla 62: C09 – Abrir conexión

Cerrar conexión	
Identificador	C10
Origen	RC10, RC13, RC14, RG01, RG03, RG04
Objetivo	Eliminar los datos de conexión e indicar al dispositivo de laboratorio que cierre las conexiones.
Acciones	
- Validar los parámetros recibidos. - Enviar mensaje al dispositivo para que finalice streaming y socket. - Eliminar la información de la conexión. - Componer la respuesta y devolverla.	

Tabla 63: C10 – Cerrar conexión

Creación agenda diaria	
Identificador	C11
Origen	RC11, RG04
Objetivo	Generar la agenda de todos los dispositivos para el periodo indicado.
Acciones	
- Guardar la agenda de todos los dispositivos que no estuviera previamente creada.	

Tabla 64: C11 – Creación agenda diaria

Consolidación reservas	
Identificador	C12
Origen	RG12, RG04
Objetivo	Archivar en el histórico las reservas vencidas.
Acciones	
- Copiar en el histórico las reservas vencidas. - Borrar la agenda pasada del dispositivo (tanto reservas vencidas como huecos libres).	

Tabla 65: C12 – Consolidación de reservas

Nuevamente la matriz de trazabilidad nos permite relacionar los requisitos de usuario esta vez con los componentes. Todos los requisitos deben verse reflejados en al menos un componente y todos los componentes deben responder al menos a un requisito de usuario.

	C01	C02	C03	C04	C05	C06	C07	C08	C09	C10	C11	C12
RC01	X											
RC02		X										
RC03			X									
RC04				X								
RC05					X							
RC06						X						
RC07							X					
RC08								X				
RC09									X			
RC10										X		
RC11											X	
RC12												X
RC13	X	X	X	X	X	X	X	X	X	X		
RC14	X	X	X	X	X	X	X	X	X	X		
RG01	X	X	X	X	X				X	X		
RG02		X										
RG03	X	X	X	X	X	X	X	X	X	X		
RG04	X	X	X	X	X	X	X	X	X	X	X	X

Tabla 66: Matriz de trazabilidad de componentes

4.3 Diseño de base de datos

MongoDB ha sido la base de datos escogida para implementar la solución. Se trata de una base de datos NoSQL orientada a documentos. Esto quiere decir que no cumple los modelos tradicionales por los que se define una estructura fija y las relaciones entre las diferentes tablas y campos de forma estable como primary keys, foreing keys, etc.

En este caso, no es natural definir una estructura cerrada puesto que por capacidad la base de datos permitiría almacenar dentro de cada collection, que es el nombre que reciben el equivalente a tablas de base de datos SQL, diferentes estructuras para cada uno de los documentos, que es el equivalente a filas de datos en las base de datos SQL, que almacena.

No obstante, en la solución sí es conocida la estructura que se va almacenar y es homogénea para todos los documentos por lo que incluimos a continuación un diagrama similar al modelo de datos.

Hay que tener en cuenta que es este tipo de base de datos es común incluir objetos o colecciones de objetos dentro de otras colecciones.

Cuando en una colección se incluye un objeto se marca con llaves ({}), cuando se incluyen colecciones se marca con corchetes ([]) y cuando son colecciones de objetos se marca con corchetes y dentro llaves ([{}]) indicando en el interior de ellos los datos que comprende.

También indicar que no se incluye en el diseño los Identificadores (_id) de las colecciones ya que se trata de un campo intrínseco de MongoDB para todas las colecciones.

Por último, no se definen tipologías para los campos ya que en MongoDB tampoco es necesario.

ApiUser

apiKey
apiSecret
description

Connection

idUser
device {
  idDevice,
  name,
  type,
  connection [{
    ip,
    port,
    description
}]}
date
start
end

Laboratory

idLab
name
description
location
available
device [{
  idDevice,
  name,
  type,
  available
  connection [{
    ip,
    port,
    description
}]}]

Reservation

idUser
device {
  idDevice,
  name,
  type,
  connection [{
    ip,
    port,
    description
}]}
laboratory{
  name
  description}
date
ende
start

Timetable

device {
  idDevice,
  name,
  type,
  available
  connection [{
    ip,
    port }]
  reservation {
    start
    end
    idReservation }
date }

TimetableHistory

device {
  idDevice,
  name,
  type,
  reservation {
    start
    end
    idReservation }
date }

Figura 11: Diagrama de base de datos

ApiUser

Esta colección almacena los credenciales de las aplicaciones finales que se conectarán al API:

- apiKey: identificador de la Aplicación consumidora.
- apiSecret: clave privada de verificación de la Aplicación consumidora.
- description: texto descriptivo que define a la Aplicación consumidora.

Connection

Esta colección almacena los datos de todas las conexiones abiertas con los dispositivos de los laboratorios:

- idUser: Identificador del usuario al que pertenece la reserva.
- device: Dispositivo con el que se ha abierto la conexión. Es un objeto que se compone de los siguientes campos:
  - idDevice: Identificador del dispositivo.
  - name: Nombre del dispositivo.
  - type: Tipo de dispositivo.
  - connection: Es un objeto de tipo connection que recoge la información necesaria para realizar la conexión con el dispositivo de laboratorio. Está compuesto por una colección de los siguientes grupos de datos:
    - ip: Ip a la que hay que realizar la conexión.
    - port: Puerto al que se debe realizar la conexión.
    - description: texto descripción del objetivo de la conexión.
- date: Fecha en la que se realiza la conexión.
- start: Hora en la que se inicia la conexión.
- end: Hora en la que se debería finalizar la conexión.

Laboratory

Esta colección almacena los datos de todos los laboratorios:

- idLab: Identificador descriptivo del laboratorio.
- name: Nombre del laboratorio.
- description: Texto descriptivo que define el tipo de laboratorio que es.
- location: Texto que describe dónde se ubica el laboratorio.
- available: Flag que determina si un laboratorio está o no disponible para reservar.
- device: Collección que recoge todos los dispositivos ubicados en el laboratorio. Cada objeto de la colección se compone de los siguientes campos:
  - idDevice: Identificador del dispositivo.
  - name: Nombre del dispositvo.
  - type: Tipo de dispositivo.

- available: Flag que indica si el dispositivo está disponible para reservas o no.
- connection: Es un objeto de tipo connection que recoge la información necesaria para realizar la conexión con el dispositivo de laboratorio. Está compuesto por una colección de los siguientes grupos de datos:
  - ip: Ip a la que hay que realizar la conexión.
  - port: Puerto al que se debe realizar la conexión.
  - description: texto descripción del objetivo de la conexión.

Reservation

Esta colección almacena los datos de todas las reservas disponibles en la para todos los dispositivos de los laboratorios:

- idUser: Identificador del usuario al que pertenece la reserva.
- device: Dispositivo que se ha reservado. Es un objeto que se compone de los siguientes campos:
  - idDevice: Identificador del dispositivo.
  - name: Nombre del dispositivo.
  - type: Tipo de dispositivo.
  - connection: Es un objeto de tipo connection que recoge la información necesaria para realizar la conexión con el dispositivo de laboratorio. Está compuesto por una colección de los siguientes grupos de datos:
    - ip: Ip a la que hay que realizar la conexión.
    - port: Puerto al que se debe realizar la conexión.
    - description: texto descripción del objetivo de la conexión.
- laboratory: laboratorio en el que está ubicado el dispositivo reservado. Se trata de un objeto compuesto de los siguientes campos:
  - name: Nombre del laboratorio.
  - description: Texto descriptivo que define el tipo de laboratorio que es.
- date: Fecha en la que se realiza la conexión.
- start: Hora en la que se inicia la conexión.
- end: Hora en la que se debería finalizar la conexión.

Timetable

Esta colección almacena la agenda de todos los dispositivos de todos los laboratorios:

- device: Dispositivo al que corresponde la agenda. Es un objeto que se compone de los siguientes campos:
  - idDevice: Identificador del dispositivo.
  - name: Nombre del dispositivo.
  - type: Tipo de dispositivo.

    - connection: Es un objeto de tipo connection que recoge la información necesaria para realizar la conexión con el dispositivo de laboratorio. Está compuesto por una colección de los siguientes grupos de datos:
        - ip: Ip a la que hay que realizar la conexión.
        - port: Puerto al que se debe realizar la conexión.
        - description: texto descripción del objetivo de la conexión.
    - reservation: Colección que almacena toda la agenda del dispositivo para la fecha dada. Se compone de los siguientes campos
        - start: hora de comienzo de la franja
        - end: hora de fin de la franja
        - idReservation: identificador de la reserva. Si el campo está vacío quiere decir que el dispositivo está disponible en esa franja.
- date: Fecha a la que corresponde la agenda.

TimetableHistory

Esta colección almacena la agenda con reservas vencidas de todos los dispositivos de todos los laboratorios:

- device: Dispositivo al que corresponde la agenda. Es un objeto que se compone de los siguientes campos:
    - idDevice: Identificador del dispositivo.
    - name: Nombre del dispositivo.
    - type: Tipo de dispositivo.
    - reservation: Colección que almacena toda la agenda del dispositivo con reservas vencidas para la fecha dada. Se compone de los siguientes campos
        - start: hora de comienzo de la franja
        - end: hora de fin de la franja
        - idReservation: identificador de la reserva.
- date: Fecha a la que corresponde la agenda.

4.4 Diagrama de clases

La solución a implementar no está estructurada propiamente en clases. No obstante, sí es posible hacer una extrapolación de la estructura del proyecto para representarla en un diagrama de clases.

El diagrama de clases nos muestra la estructura de código de la solución y las principales variables y métodos que lo componen.

A parte de las clases incluidas en el diagrama hay que tener en cuenta que también se incluyen en la solución dos procesos batch:

- timetable → Que genera la agenda diaria

- timetable_history → Que archiva las reservas vencidas de la agenda y libera todas las entradas pasadas (con o sin reserva) de la agenda.

Estos dos procesos batch están compuestos de una clase cada uno que se ejecuta de manera independiente de la solución principal.

La solución principal implementa el API y sus métodos y está compuesta por las siguientes clases:

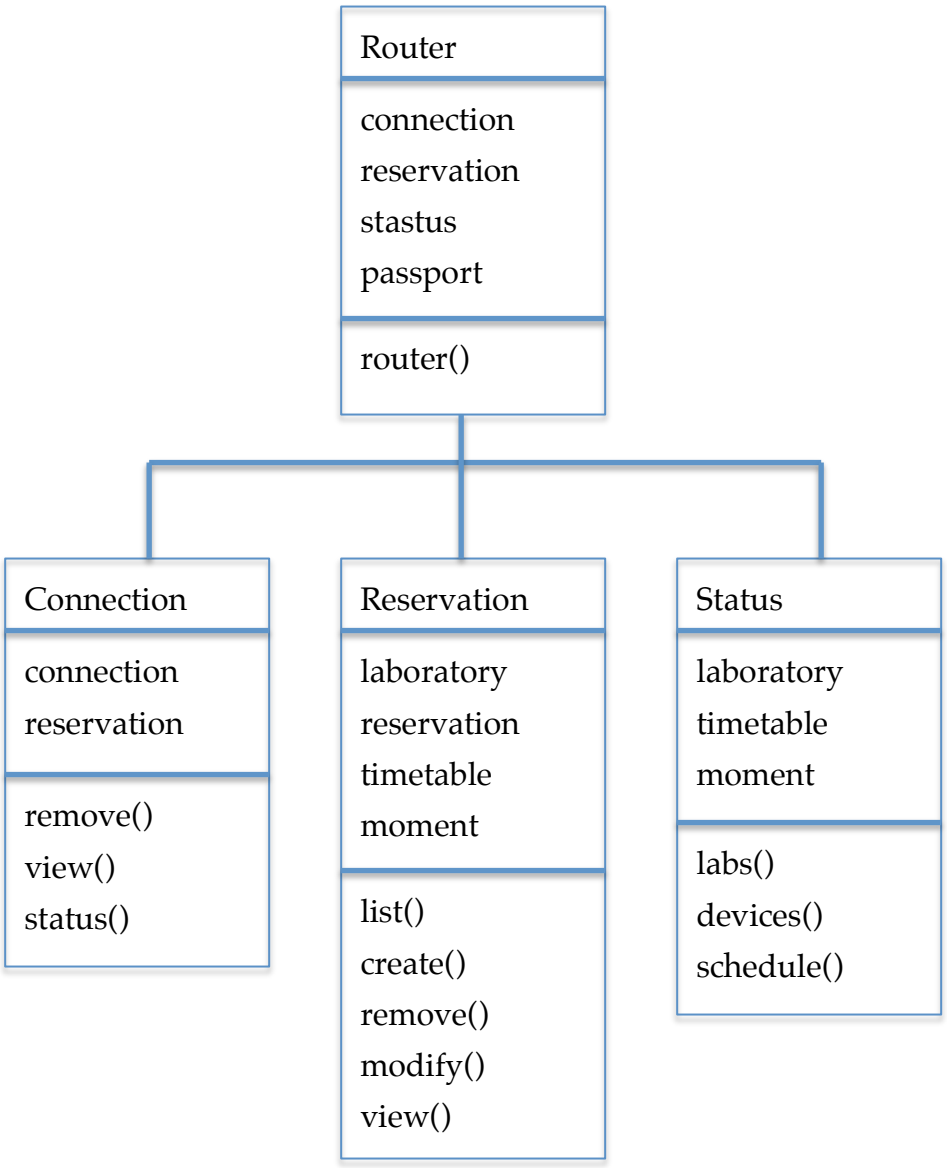


Figura 12: Diagrama de clases

4.5 Diseño de interfaces

En esta solución no hay un interfaz gráfico. No obstante, sí hay un interfaz definido para el uso del API que se puede extrapolar para aplicar en este apartado.

Para la definición de dicho interfaz API se ha utilizado RAML (http://raml.org). RAML es el acrónimo de RESTful API Modeling Language. Su propósito principal es la definición de APIs REST de manera entendible para las personas. Está basado en YAML (http://yamel.org).

Para la documentación en concreto se ha utilizado la herramienta Mulesoft que es de uso gratuito y que además tiene la ventaja de que es online, por lo que es mucho más accesible que las herramientas tradicionales de documentación.

Se puede acceder a la definición completa del interfaz del proyecto en la siguiente dirección:
https://anypoint.mulesoft.com/apiplatform/beeva/#/portals/organizations/75abf700-88a7-42d7-9bdd-ae42040809bb/apis/9989/versions/10289/pages/8276

En el Anexo B de este documento se incluye la documentación completa generada en RAML para esta solución.

5 Implementación

5.1 Herramientas desarrollo

Las herramientas utilizadas para la implementación han sido todas de software libre o propias del sistema operativo utilizado . El listado completo de herramientas utilizadas son:

- **Node.js** (https://nodejs.org/)→ Como servidor que recibe las peticiones API REST desde las aplicaciones consumidoras y también como cliente REST para invocar a las interfaces de los dispositivos en sus arranques y paradas.

- **npm** (https://www.npmjs.com/)→ Gestor de dependencias de librerías para Node.js, que permite la descarga e instalación de todas las dependencias del proyecto centralizadas desde un solo archivo: package.json

- **MongoDB** (https://www.mongodb.org/) → Como base de datos NoSQL para almacenar todos los datos de laboratorios y dispositivos, así como las agendas futuras y las reservas, tanto pasadas como futuras, realizadas sobre los dispositivos.

- **Atom** (https://atom.io/) → Como entorno de desarrollo, por su capacidad de integración con lenguaje Javascript (Node.js) además de su alta cantidad de plugins que simplifican el desarrollo.

- **iTerm2** (https://www.iterm2.com/)→ Consola o terminal avanzada en entornos Mac OS que permite la división automática de ventanas dentro de un mismo terminal. Utilizada para realizar varias tareas como: ejecución de la aplicación, consulta de los datos en MongoDB, creación de los claves utilizadas para la comunicación SSL…

- **API Plaftorm de Mulesoft** (https://www.mulesoft.com/)→ Herramienta web utilizada para crear la especificación RAML de los servicios implementados en el API.

- **Postman** (https://www.getpostman.com/) → Aplicación que permite realizar peticiones HTTP de todo tipo, pudiendo así probar los servicios del API de forma unitaria.

5.2 Puntos claves de la implementación

La implementación del proyecto es la fase en la que se codifica la solución siguiendo el diseño técnico definido y cubriendo los requisitos de software definidos a partir de los requisitos de usuario.

Los puntos claves de la implementación han sido:

- Generación de claves para el uso de SSL en la comunicación con el backend e implementación de interceptor de autenticación bajo el estándar OAuth 1.0

- Selección de plataforma para la especificación de los servicios del API así como su posterior implementación. De nada sirve generar un backend de servicios REST si no hay una documentación asociada que indique de que manera se utiliza.

- Definición del esquema a utilizar en el modelo de datos a utilizar en MongoDB. En un inicio parece que todos los datos tenderían a estar relacionados y que nos llevan a un modelo relacional pero en realidad se puede hacer uso de la potencia de anidamiento de los documentos en MongoDB para minimizar el impacto a la hora de acceder a los datos desde los distintos servicio, a pesar de que aumenta el tamaño usado para el almacenamiento.

6 Pruebas

Las pruebas se realizan desde que comienza la implementación hasta el final del proyecto. Según el momento, se realizan pruebas de diferentes tipologías: desde las unitarias que son las pruebas que realiza el técnico durante el desarrollo, las pruebas de aceptación por parte del usuario o las pruebas de regresión una vez se ha puesto en producción el sistema. Cada una de las pruebas tiene una finalidad y una metodología.

A continuación se incluye las pruebas funcionales, puesto que son aquellas que tienen como finalidad garantizar que los requisitos de software, y por tanto los requisitos de usuario, han sido correctamente implementados.

6.1 Configuraciones

Las pruebas funcionales han de ser realizadas en un entorno controlado en el que podamos saber cual es el comportamiento esperado del sistema. Para ello es necesario tener clara la configuración de datos de la que se parte.

En este caso vamos a utilizar los siguientes datos de partida:

- Usuario válido de la UC3M: 100033868@alumnos.uc3m.es
    - Ninguna reserva dada de alta en el sistema para este usuario
- Fecha del sistema: 01/10/2015
- Laboratorio 1:
    - Agenda: todos los horarios disponibles incluido hasta el día 08/10/2015 23:30 – 00:00:
    - idLab: LAB_004
    - name: 4.1.C02
    - description: Laboratorio genérico
    - location: Edificio Torres Quevedo
    - available: true
    - devices:
        - Device1:
            - type: Motor
            - available: true
            - idDev: DEV_004_001
            - name: Motor 4.001
            - connection:
                - IP: 185.34.79.23
                - port: 3456
        - Device 2:
            - type: Circuito
            - available: true
            - idDev: DEV_004_002
            - name: Circuito 4.002
            - connection:

        - IP: 185.34.79.24
        - port: 3456
    - Device 3:
        - type: Autómata
        - available: false
        - idDev: DEV_004_003
        - name: Autómata 4.003
        - connection:
            - IP: 185.34.79.25
            - port: 3456
- Laboratorio 2:
    - idLab: LAB_002
    - name: 2.2.B02
    - description: Laboratorio cerrado
    - location: Edificio Sabatini
    - available: false
    - devices: No devices

Todas las pruebas se entienden secuenciales en el orden indicado puesto que las acciones realizadas en una son base para probar las siguientes.

Se podría realizar una configuración de datos para las pruebas de modo que no hiciera falta que lo fueran. Para ello se podría utilizar una de las dos siguientes técnicas:

- Diferentes escenarios de configuración de pruebas
- Un único escenario de configuración pero con un conjunto de datos mucho más amplio de datos de modo que las modificaciones generadas por unas pruebas no afectaran a los grupos de datos de las otras pruebas.

6.2 Pruebas de funcionalidad

PF01 – Crear nueva reserva	
Se recibe una llamada al método del API "Crear nueva reserva".	
Requisitos	RF01, RNF01
Resultado	Correcto
Pasos a seguir	

- Recibida llamada al método "Crear nueva reserva" del API con los siguientes parámetros:
  - idDev: DEV_004_001 → En la URL
  - idUser: 100033868@alumnos.uc3m.es
  - date: 02/10/2015
  - start: 12:00
  - end: 12:30
- El sistema devuelve OK como respuesta.
- Comprobamos que en la base de datos se ha generado la reserva correctamente.
- Recibida segunda llamada al método "Crear nueva reserva" del API con los mismo parámetros que en el caso contrario:
  - idDev: DEV_004_001 → En la URL
  - idUser: 100033868@alumnos.uc3m.es
  - date: 02/10/2015
  - start: 12:00
  - end: 12:30
- El sistema devuelve ERROR como respuesta. Motivo: dispositivo ya ocupado en esa hora.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida tercera llamada al método "Crear nueva reserva" del API con los siguientes parámetros:
  - idDev: DEV_004_001 → En la URL
  - idUser: 100033868@alumnos.uc3m.es
  - date: 05/10/2015
  - start: 12:00
  - end: 12:30
- El sistema devuelve OK como respuesta.
- Comprobamos que en la base de datos se ha generado la reserva correctamente.
- Recibida cuarta llamada al método "Crear nueva reserva" del API con

los siguientes parámetros:
    - idDev: DEV_004_002 → En la URL
    - idUser: 100033868@alumnos.uc3m.es
    - date: 05/10/2015
    - start: 12:00
    - end: 12:30
- El sistema devuelve ERROR como respuesta. Motivo: este usuario ya tiene una reserva para la misma fecha y hora.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida quinta llamada al método "Crear nueva reserva" del API con los siguientes parámetros:
    - idDev: DEV_004_002 → En la URL
    - idUser: 100033868@alumnos.uc3m.es
    - date: 05/10/2015
    - start: 14:00
    - end: 14:30
- El sistema devuelve OK como respuesta.
- Comprobamos que en la base de datos se ha generado la reserva correctamente.
- Recibida sexta llamada al método "Crear nueva reserva" del API con los siguientes parámetros:
    - idDev: DEV_005_001 → En la URL
    - idUser: 100033868@alumnos.uc3m.es
    - date: 03/10/2015
    - start: 12:00
    - end: 12:30
- El sistema devuelve ERROR como respuesta. Motivo: este dispositivo no existe.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida séptima llamada al método "Crear nueva reserva" del API con los siguientes parámetros:
    - idDev: DEV_004_003 → En la URL
    - idUser: 100033868@alumnos.uc3m.es
    - date: 03/10/2015
    - start: 12:00
    - end: 12:30
- El sistema devuelve ERROR como respuesta. Motivo: este dispositivo no está disponible para reservar.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida octava llamada al método "Crear nueva reserva" del API con los siguientes parámetros:
    - idDev: DEV_004_001 → En la URL
    - idUser: 100033868@alumnos.uc3m.es
    - date: 03/09/2015
    - start: 12:00

  - end: 12:30
- El sistema devuelve ERROR como respuesta. Motivo: Fecha anterior al día actual.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida novena llamada al método "Crear nueva reserva" del API con los siguientes parámetros:
  - idDev: DEV_004_003 → En la URL
  - idUser: 100033868@alumnos.uc3m.es
  - date: 03/11/2015
  - start: 12:00
  - end: 12:30
- El sistema devuelve ERROR como respuesta. Motivo: Agenda no disponible para esa fecha.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida decima llamada al método "Crear nueva reserva" del API sin parámetros.
- El sistema devuelve ERROR como respuesta. Motivo: La llamada no incluye los parámetros requeridos.
- Comprobamos que en la base de datos no se ha producido modificación.

Tabla 67: PF01 - Crear nueva reserva

PF02 – Modificar reserva	
Se recibe una llamada al método del API "Modificar reserva".	
Requisitos	RF02, RNF02
Resultado	Correcto

Pasos a seguir

- Se consulta en la base de datos las reservas disponibles para el usuario 100033868@alumnos.uc3m.es
  - Se comprueba que este usuario tiene 3 reservas disponibles.
  - Se toma el idReserva de la reserva con los siguientes datos:
    - idDev: DEV_004_001
    - idUser: 100033868@alumnos.uc3m.es
    - date: 02/10/2015
    - start: 12:00
    - end: 12:30
- Recibida llamada al método "Modificar reserva" del API con los siguientes parámetros:
  - idReserva: Valor que recogimos antes de la base de datos → En la URL
  - idDev: DEV_004_001
  - idUser: 100033868@alumnos.uc3m.es
  - date: 02/10/2015
  - start: 12:00
  - end: 12:30
- El sistema devuelve OK como respuesta.
- Comprobamos que en la base de datos que se mantienen los mismos datos que ya había anteriormente registrados.
- Recibida segunda llamada al método "Modificar reserva" del API con los siguientes parámetros:
  - idReserva: Valor que recogimos antes de la base de datos → En la URL
  - idDev: DEV_004_001
  - idUser: 100033868@alumnos.uc3m.es
  - date: 02/10/2015
  - start: 14:00
  - end: 14:30
- El sistema devuelve OK como respuesta.
- Comprobamos que en la base de datos se ha modificado la reserva.
- Recibida tercera llamada al método "Modificar reserva" del API con los siguientes parámetros:

    - idReserva: Valor que recogimos antes de la base de datos → En la URL
    - idDev: DEV_004_001
    - idUser: 100033868@alumnos.uc3m.es
    - date: 05/10/2015
    - start: 12:00
    - end: 12:30
- El sistema devuelve ERROR como respuesta. Motivo: el usuario ya tiene reserva hecha para ese mismo horario.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida cuarta llamada al método "Modificar reserva" del API con los mismos parámetros que en la segunda llamada:
    - idReserva: Valor que recogimos antes de la base de datos → En la URL
    - idDev: DEV_004_001
    - idUser: 100033868@alumnos.uc3m.es
    - date: 02/10/2015
    - start: 14:00
    - end: 14:30
- El sistema devuelve ERROR como respuesta. Motivo: dispositivo ya ocupado en esa hora.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida quinta llamada al método "Modificar reserva" del API con los siguientes parámetro:
    - idReserva: Valor que recogimos antes de la base de datos → En la URL
    - idDev: DEV_004_001
    - idUser: 100033868@alumnos.uc3m.es
    - date: 05/10/2015
    - start: 12:00
    - end: 12:30
- El sistema devuelve ERROR como respuesta. Motivo: este usuario ya tiene una reserva para la misma fecha y hora.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida sexta llamada al método "Modificar reserva" del API con los siguientes parámetro:
    - idReserva: Valor que recogimos antes de la base de datos → En la URL
    - idDev: DEV_005_001
    - idUser: 100033868@alumnos.uc3m.es
    - date: 03/10/2015
    - start: 12:00
    - end: 12:30
- El sistema devuelve ERROR como respuesta. Motivo: este dispositivo no existe.

- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida séptima llamada al método "Modificar reserva" del API con los siguientes parámetro:
  - idReserva: Valor que recogimos antes de la base de datos → En la URL
  - idDev: DEV_004_003
  - idUser: 100033868@alumnos.uc3m.es
  - date: 03/10/2015
  - start: 12:00
  - end: 12:30
- El sistema devuelve ERROR como respuesta. Motivo: este dispositivo no está disponible para reservar.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida octava llamada al método "Modificar reserva" del API con los siguientes parámetro:
  - idReserva: Valor que recogimos antes de la base de datos → En la URL
  - idDev: DEV_004_001
  - idUser: 100033868@alumnos.uc3m.es
  - date: 03/09/2015
  - start: 12:00
  - end: 12:30
- El sistema devuelve ERROR como respuesta. Motivo: Fecha anterior al día actual.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida novena llamada al método "Modificar reserva" del API con los siguientes parámetro:
  - idReserva: Valor que recogimos antes de la base de datos → En la URL
  - idDev: DEV_004_001
  - idUser: 100033868@alumnos.uc3m.es
  - date: 03/11/2015
  - start: 12:00
  - end: 12:30
- El sistema devuelve ERROR como respuesta. Motivo: Agenda no disponible para esa fecha.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida decima llamada al método "Modificar reserva" del API sin parámetros.
- El sistema devuelve ERROR como respuesta. Motivo: La llamada no incluye los parámetros requeridos.
- Comprobamos que en la base de datos no se ha producido modificación.

Tabla 68: PF02 - Modificar reserva

PF03 – Eliminar reserva	
Se recibe una llamada al método del API "Eliminar reserva".	
Requisitos	RF03, RNF03
Resultado	Correcto
Pasos a seguir	

- Se consulta en la base de datos las reservas disponibles para el usuario 100033868@alumnos.uc3m.es
  - Se comprueba que este usuario tiene 3 reservas disponibles.
  - Se toma el idReserva de la reserva con los siguientes datos:
    - idDev: DEV_004_002
    - idUser: 100033868@alumnos.uc3m.es
    - date: 05/10/2015
    - start: 14:00
    - end: 14:30
- Recibida llamada al método "Eliminar reserva" del API con los siguientes parámetros:
  - idReserva: Valor que recogimos antes de la base de datos → En la URL
- El sistema devuelve OK como respuesta.
- Comprobamos que en la base de datos que se mantienen los mismos datos que ya había anteriormente registrados.
- Recibida segunda llamada al método "Eliminar reserva" del API con los mismos parámetros que en la primera llamada:
  - idReserva: Valor que recogimos antes de la base de datos → En la URL
- El sistema devuelve ERROR como respuesta. Motivo: la reserva no existe.
- Comprobamos que en la base de datos no se ha producido modificación.
- Cambiamos la fecha del servidor para que sea 03/10/2015.
- Se consulta en la base de datos las reservas disponibles para el usuario 100033868@alumnos.uc3m.es
  - Se comprueba que este usuario tiene 2 reservas almacenadas, 1 de ellas no vencida .
  - Se toma el idReserva de la reserva con los siguientes datos:
    - idDev: DEV_004_001
    - idUser: 100033868@alumnos.uc3m.es
    - date: 02/10/2015
    - start: 14:00
    - end: 14:30

- Recibida tercera llamada al método "Eliminar reserva" del API con los siguientes datos:
  - idReserva: Valor que recogimos en esta última consulta a base de datos → En la URL
- El sistema devuelve ERROR como respuesta. Motivo: la reserva ya ha vencido.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida cuarta llamada al método "Eliminar reserva" del API sin parámetros.
- El sistema devuelve ERROR como respuesta. Motivo: La llamada no incluye los parámetros requeridos.
- Comprobamos que en la base de datos no se ha producido modificación.

Tabla 69: PF03 - Eliminar reserva

PF04 – Recuperar detalle de la reserva	
Se recibe una llamada al método del API "Eliminar reserva".	
Requisitos	RF04, RNF04
Resultado	Correcto

Pasos a seguir

- Se consulta en la base de datos las reservas disponibles para el usuario 100033868@alumnos.uc3m.es
  - Se comprueba que este usuario tiene 2 reservas almacenadas, 1 de ellas no vencida.
  - Se toma el idReserva de la reserva con los siguientes datos:
    - idDev: DEV_004_001 → En la URL
    - idUser: 100033868@alumnos.uc3m.es
    - date: 05/10/2015
    - start: 12:00
    - end: 12:30
- Recibida llamada al método "Recuperar detalle reserva" del API con los siguientes datos:
  - idReserva: Valor que recogimos en esta última consulta a base de datos → En la URL
- El sistema devuelve los siguientes datos como respuesta:
  - idUser: 100033868@alumnos.uc3m.es
  - date: 05/10/2015
  - start: 12:00
  - end: 12:30
  - laboratory:
    - name: 4.1.C02
    - description: Laboratorio genérico
  - device:
    - type: Motor
    - idDev: DEV_004_001
    - name: Motor 4.001
    - connection:
      - IP: 185.34.79.23
      - port: 3456
- Comprobamos que en la base de datos no se ha producido modificación.
- Se consulta en la base de datos las reservas disponibles para el usuario 100033868@alumnos.uc3m.es
  - Se comprueba que este usuario tiene 2 reservas almacenadas, 1 de ellas no vencida .

    - Se toma el idReserva de la reserva con los siguientes datos:
        - idDev: DEV_004_001
        - idUser: 100033868@alumnos.uc3m.es
        - date: 02/10/2015
        - start: 14:00
        - end: 14:30
- Recibida segunda llamada al método "Recuperar detalle reserva" del API con los siguientes datos:
    - idReserva: Valor que recogimos en esta última consulta a base de datos → En la URL
- El sistema devuelve ERROR como respuesta. Motivo: la reserva ya ha vencido.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida tercera llamada al método "Recuperar detalle reserva" del API con los siguientes datos:
    - idReserva: Valor inventado no correspondiente a ninguna reserva de la base de datos → En la URL
- El sistema devuelve ERROR como respuesta. Motivo: la reserva no existe.
- Comprobamos que en la base de datos no se ha producido modificación.
- Recibida cuarta llamada al método "Recuperar detalle reserva" del API sin parámetros.
- El sistema devuelve ERROR como respuesta. Motivo: La llamada no incluye los parámetros requeridos.
- Comprobamos que en la base de datos no se ha producido modificación.

Tabla 70: PF04 - Recuperar detalle de la reserva

PF05 – Recuperar reservas de usuario	
Se recibe una llamada al método del API "Recuperar reservas de usuario".	
Requisitos	RF05, RNF05
Resultado	Correcto
Pasos a seguir	

- Recibida llamada al método "Recuperar reservas de usuario" del API con los siguientes datos:
  - idUser: 10033868@uc3m.es → En la URL
- El sistema devuelve como respuesta 1 reserva con los siguientes datos:
  - idUser: 100033868@alumnos.uc3m.es
  - date: 05/10/2015
  - start: 12:00
  - end: 12:30
  - laboratory:
    - name: 4.1.C02
    - description: Laboratorio genérico
  - device:
    - type: Motor
    - idDev: DEV_004_001
    - name: Motor 4.001
    - connection:
      - IP: 185.34.79.23
      - port: 3456
  - datetime: 05/10/2015 12:00
- Recibida segunda llamada al método "Recuperar reservas de usuario" del API con los siguientes datos:
  - idUser: prueba@uc3m.es → En la URL
- El sistema un resultado vacío como respuesta pero no devuelve error. Motivo: el usuario no tiene reservas en el sistema.
- Recibida tercera llamada al método "Recuperar reservas de usuario" del API sin parámetros.
- El sistema devuelve ERROR como respuesta. Motivo: La llamada no incluye los parámetros requeridos.
- Comprobamos que en la base de datos no se ha producido modificación.

Tabla 71: PF05 - Recuperar reservas de usuario

PF06 – Buscar laboratorios	
Se recibe una llamada al método del API "Buscar laboratorios".	
Requisitos	RF06, RNF06
Resultado	Correcto
Pasos a seguir	

- Recibida llamada al método "Buscar laboratorios" del API sin parámetros.
- El sistema devuelve como respuesta los siguientes datos
  - idLab: LAB_004
  - name: 4.1.C02
  - description: Laboratorio genérico
  - location: Edificio Torres Quevedo
  - available: true
  - devices:
    - Device1:
      - type: Motor
      - available: true
      - idDev: DEV_004_001
      - name: Motor 4.001
      - connection:
        - IP: 185.34.79.23
        - port: 3456
    - Device 2:
      - type: Circuito
      - available: true
      - idDev: DEV_004_002
      - name: Circuito 4.002
      - connection:
        - IP: 185.34.79.24
        - port: 3456

Tabla 72: PF06 - Buscar laboratorios

PF07 – Buscar dispositivos	
Se recibe una llamada al método del API "Buscar dispositivos".	
Requisitos	RF07, RNF07
Resultado	Correcto

Pasos a seguir

- Recibida llamada al método "Buscar dispositivos" del API sin parámetros.
- El sistema devuelve como respuesta los siguientes datos que se corresponden con todos los dispositivos disponibles:
  - idLab: LAB_004
  - name: 4.1.C02
  - description: Laboratorio genérico
  - location: Edificio Torres Quevedo
  - available: true
  - devices:
    - Device1:
      - type: Motor
      - available: true
      - idDev: DEV_004_001
      - name: Motor 4.001
      - connection:
        - IP: 185.34.79.23
        - port: 3456
    - Device 2:
      - type: Circuito
      - available: true
      - idDev: DEV_004_002
      - name: Circuito 4.002
      - connection:
        - IP: 185.34.79.24
        - port: 3456
- Recibida segunda llamada al método "Buscar dispositivos" del API con los siguientes parámetros en la URL.
  - idLab: LAB_004
- El sistema devuelve como respuesta los siguientes datos que se corresponden con todos los dispositivos del laboratorio indicado:
  - idLab: LAB_004
  - name: 4.1.C02

    - description: Laboratorio genérico
    - location: Edificio Torres Quevedo
    - available: true
    - devices:
        - Device1:
            - type: Motor
            - available: true
            - idDev: DEV_004_001
            - name: Motor 4.001
            - connection:
                - IP: 185.34.79.23
                - port: 3456
        - Device 2:
            - type: Circuito
            - available: true
            - idDev: DEV_004_002
            - name: Circuito 4.002
            - connection:
                - IP: 185.34.79.24
                - port: 3456
- Recibida tercera llamada al método "Buscar dispositivos" del API con los siguientes parámetros en la URL.
    - type: Circuito
- El sistema devuelve como respuesta los siguientes datos que se corresponden con todos los dispositivos del tipo "Circuito":
    - idLab: LAB_004
    - name: 4.1.C02
    - description: Laboratorio genérico
    - location: Edificio Torres Quevedo
    - available: true
    - devices:
        - Device 1:
            - type: Circuito
            - available: true
            - idDev: DEV_004_002
            - name: Circuito 4.002
            - connection:
                - IP: 185.34.79.24
                - port: 3456
- Recibida cuarta llamada al método "Buscar dispositivos" del API con los siguientes parámetros en la URL.
    - idDev: DEV_004_002
- El sistema devuelve como respuesta los siguientes datos que se corresponden con el dispositivo concreto solicitado:

    - idLab: LAB_004
    - name: 4.1.C02
    - description: Laboratorio genérico
    - location: Edificio Torres Quevedo
    - available: true
    - devices:
        - Device 1:
            - type: Circuito
            - available: true
            - idDev: DEV_004_002
            - name: Circuito 4.002
            - connection:
                - IP: 185.34.79.24
                - port: 3456
- Recibida quinta llamada al método "Buscar dispositivos" del API con los siguientes parámetros en la URL.
    - idDev: DEV_004_002
    - idLab: LAB_002
    - type: Autómata
- El sistema devuelve como respuesta los siguientes datos que se corresponden con el dispositivo concreto solicitado ya que dicho parámetro prevalece sobre el de idLab o type:
    - idLab: LAB_004
    - name: 4.1.C02
    - description: Laboratorio genérico
    - location: Edificio Torres Quevedo
    - available: true
    - devices:
        - Device 1:
            - type: Circuito
            - available: true
            - idDev: DEV_004_002
            - name: Circuito 4.002
            - connection:
                - IP: 185.34.79.24
                - port: 3456
- Recibida sexta llamada al método "Buscar dispositivos" del API con los siguientes parámetros en la URL.
    - idDev: DEV_004_002
- El sistema un resultado vacío como respuesta pero no devuelve error. Motivo: el tipo de dispositivo no existe.
- Recibida séptima llamada al método "Buscar dispositivos" del API con los siguientes parámetros en la URL.
    - idLab: LAB_005

• El sistema un resultado vacío como respuesta pero no devuelve error. Motivo: el laboratorio no existe. • Recibida octava llamada al método "Buscar dispositivos" del API con los siguientes parámetros en la URL.   o type: lamp • El sistema un resultado vacío como respuesta pero no devuelve error. Motivo: el tipo de dispositivo no existe.

Tabla 73: PF07 - Buscar dispositivos

PF08 – Recuperar agenda de dispositivo	
Se recibe una llamada al método del API "Recuperar agenda de dispositivo".	
Requisitos	RF08, RNF08
Resultado	Correcto
Pasos a seguir	

• Recibida llamada al método "Recuperar agenda de dispositivo" del API con los siguientes parámetros:
  o idDev: DEV_004_001 → En la URL
• El sistema devuelve como respuesta la agenda disponible del dispositivo siguiendo la siguiente estructura:
  o date → una colección por cada día disponible.
  ▪ times → que incluye todos los rangos disponibles que se componen de:
  • start
  • end
• Comprobar que para este dispositivo están todas las fechas no vencidas disponibles hasta (incluido) 08/10/2015 23:30 – 00:00.
• Recibida segunda llamada al método "Recuperar agenda de dispositvo" del API con los siguientes parámetros:
  o idDev: DEV_005_001 → En la URL
• El sistema un resultado vacío como respuesta pero no devuelve error. Motivo: el tipo de dispositivo no existe.	

Tabla 74: PF08 - Recuperar agenda de dispositivo

PF09 – Abrir conexión	
Se recibe una llamada al método del API "Recuperar agenda de dispositivo".	
Requisitos	RF09, RNF09
Resultado	Correcto
Pasos a seguir	

- Se consulta en la base de datos las reservas disponibles para el usuario 100033868@alumnos.uc3m.es
  - Se comprueba que este usuario tiene 2 reservas almacenadas, 1 de ellas no vencida.
  - Se toma el idReserva de la reserva con los siguientes datos:
    - idDev: DEV_004_001 → En la URL
    - idUser: 100033868@alumnos.uc3m.es
    - date: 05/10/2015
    - start: 12:00
    - end: 12:30
- Cambiar fecha del servidor y el ordenador de las pruebas a 05/10/2015 a las 12:10.
- Recibida llamada al método "Abrir conexión" del API con los siguientes parámetros:
  - idReserva: Valor que recogimos en esta última consulta a base de datos → En la URL
- El sistema devuelve como respuesta los siguientes datos:
  - idConexión
  - idUser: 100033868@alumnos.uc3m.es
  - date: 05/10/2015
  - start: 12:00
  - end: 12:30
  - device:
    - type: Motor
    - idDev: DEV_004_001
    - name: Motor 4.001
    - connection:
      - IP: 185.34.79.23
      - port: 3456
- Comprobar que el sistema ha lanzado una llamada al dispositivo de laboratorio indicándole que arranque.
- Comprobar que el sistema ha guardado en la tabla Connections el registro correspondiente al id de conexión.
- Recibida segunda llamada al método "Abrir conexión" del API con los

siguientes parámetros:
    - idReserva: valor inventado que no corresponde a ninguna reserva → En la URL
- El sistema devuelve ERROR como respuesta. Motivo: Se está intentando acceder a una reserva que no existe.
- Comprobamos que en la base de datos no se ha producido modificación.

Tabla 75: PF09 - Abrir conexión

PF10 – Cerrar conexión	
Se recibe una llamada al método del API "Recuperar agenda de dispositivo".	
Requisitos	RF10, RNF10
Resultado	Correcto
Pasos a seguir	

- Se consulta en la base de datos las conexiones disponibles para el usuario 100033868@alumnos.uc3m.es
    - Se comprueba que este usuario tiene 1 activa.
    - Se toma el idConexion de la reserva con los siguientes datos:
        - idDev: DEV_004_001 → En la URL
        - idUser: 100033868@alumnos.uc3m.es
        - date: 05/10/2015
        - start: 12:00
        - end: 12:30
- Recibida llamada al método "Cerrar conexión" del API con los siguientes parámetros:
    - idConexion: Valor que recogimos en esta última consulta a base de datos → En la URL
- El sistema devuelve OK como respuesta.
- Comprobar que el sistema ha lanzado una llamada al dispositivo de laboratorio indicándole que pare.
- Comprobar que el sistema ha eliminado en la tabla Connections el registro con el id de conexión.
- Recibida segunda llamada al método "Cerrar conexión" del API con los siguientes parámetros:
    - idConnection: valor inventado que no corresponde a ninguna conexión → En la URL
- El sistema devuelve ERROR como respuesta. Motivo: Se está intentando

cerrar una conexión que no existe. • Comprobamos que en la base de datos no se ha producido modificación.

Tabla 76: PF10 - Cerrar conexión

PF11 – Creación agenda diaria	
Se ejecuta el proceso "Creación agenda diaria".	
Requisitos	RF11
Resultado	Correcto
Pasos a seguir	
• Ejecutar el proceso "Creación agenda diaria". • Comprobar que el proceso termina sin errores. • Confirmar en base de datos que se han creado todos los registros de agenda que faltaban hasta el 12/10/2015 en la franja (incluida) 12:00 – 12:30. Esto es debido a que la configuración del proceso está a 7 días. • Ejecutar nuevamente el proceso "Creación agenda diaria". • Comprobar que el proceso termina sin errores. • Confirmar que no ha habido modificaciones en base de datos.	

Tabla 77: PF11 - Creación agenda diaria

PF12 – Consolidación de agenda vencida	
Se ejecuta el proceso "Consolidación de agenda vencida".	
Requisitos	RF12
Resultado	Correcto
Pasos a seguir	
• Ejecutar el proceso "Consolidación de agenda vencida". • Comprobar que el proceso termina sin errores. • Confirmar en base de datos que se ha eliminado la agenda anterior a	

05/10/2015 12:00.

- Confirmar en base de datos que se ha movido al histórico la reserva con datos:
  - idDev: DEV_004_001
  - idUser: 100033868@alumnos.uc3m.es
  - date: 02/10/2015
  - start: 14:00
  - end: 14:30
- Ejecutar nuevamente el proceso "Consolidación de agenda vencida".
- Comprobar que el proceso termina sin errores.
- Confirmar que no ha habido modificaciones en base de datos.

Tabla 78: PF12 - Consolidación de agenda vencida

6.4 Matriz de trazabilidad

La matriz de trazabilidad relaciona los requisitos de software con las pruebas funcionales.

Cada requisito de software debe tener, al menos, una prueba funcional. Cada prueba funcional debe comprobar el correcto comportamiento de 1 o más requisitos de software.

	RF01	RF02	RF03	RF04	RF05	RF06	RF07	RF08	RF09	RF10	RF11	RF12	RNF01	RNF02	RNF03	RNF04	RNF05	RNF06	RNF07	RNF08	RNF09	RNF10
PF01	X												X									
PF02		X												X								
PF03			X												X							
PF04				X												X						
PF05					X												X					
PF06						X												X				
PF07							X												X			
PF08								X												X		
PF09									X												X	
PF10										X												X
PF11											X											
PF12												X										

7 Conclusiones

A la finalización del proyecto vamos a remarcar las conclusiones extraídas tras la realización del proyecto así como las posibles líneas de desarrollo que se abren una vez finalizado el mismo, a tratar como cambios, mejoras o nuevas especificaciones que no se habían tenido en cuenta al inicio del proyecto.

7.1 Conclusiones del proyecto

Este proyecto debería de suponer el comienzo de una línea de trabajo a seguir en futuros avances: implementación independiente de cada pieza que compone el proyecto, división de tareas, punto único de entrada al sistema y tratamiento generalizado a través de interceptores o aspectos. Cualquier modificación que se tenga que realizar desde este momento no debería de suponer ningún cambio para la funcionalidad o funcionalidades con las que no interfiera, lo que dota al sistema de desarrollo de una gran potencia y ventaja frente a sistemas donde cada pieza forma parte de un todo.

Además, se ha trabajado con nuevas herramientas de desarrollo, como son Node.js o MongoDB, que pese a llevar poco tiempo con nosotros ya cuentan con una gran comunidad a sus espaldas lo que permite tener un rápido y fiable feedback de todo el ecosistema, así como un conjunto de respuestas extenso en los foros y grupos de debates que permiten a los desarrolladores avanzar de forma mucho más ágil.

Por todo ello, los objetivos con lo que se comenzó el proyecto quedarían cubiertos a la finalización del mismo, dando la posibilidad de realizar futuros desarrollos que amplíen la funcionalidad implementada hasta el momento.

7.2 Conclusiones personales

En el apartado personal, este proyecto ha supuesto poner en marcha un conjunto de herramientas y tecnologías punteras en la actualidad, ponerse al día con los sistemas de autenticación y sus diferentes implementaciones para poder hacer uso de ellas en la aplicación desarrollada y ver la solución generada como el componente de un todo y no como un sistema independiente, lo cual siempre ayuda a tener un visión más amplia de las problemáticas expuestas.

También cabe remarcar que produce cierta ilusión saber que este proyecto podrá se utilizado por futuros alumnos de la Universidad y que posiblemente les hará que su día a día con ciertas asignaturas sea un poco más sencillo. De esta manera, intento devolver todo lo que se me ha dado estos años y que han hecho posible mi evolución como profesional.

7.3 Mejoras futuras

Este proyecto está ideado en si mismo como el comienzo de un sistema final mucho mayor. Por tanto las líneas de mejora futuras son muy amplias.

Los cambios principales que se proponen para mejorar el proyecto son los siguientes:

- Sistema backend para la gestión de los datos que componen el sistema: laboratorios, dispositivos, agenda, tipos de práctica y aplicaciones que tendrán acceso al sistema. Facilitando así la administración y mantenimiento de la aplicación.
- Generación automática de tests que permitan probar la aplicación de forma automatizada en un entorno de integración continua. De esta forma se gana tiempo a la hora de crear o modificar funcionalidades y se evitan que salgan nuevo errores sobre funcionalidades ya probadas anteriormente.
- Gestión de errores basada en códigos de estado de HTTP, de forma que el error no pertenezca al cuerpo del mensaje de la respuesta sino que directamente venga especificado en la cabecera HTTP.
- Sistema de monitorización del API que reaccione ante posibles problemas. El objetivo sería conseguir un sistema de alta disponibilidad con la mínima intervención humana.
- Sistema de notificación (vía mail o SMS) para que los administradores estén informados de los errores que se hayan podido producir, las acciones tomadas automáticamente y el resultado obtenido de las mismas.
- Sistema de notificación (vía mail o SMS) para que los estudiantes, que les emitan recordatorios de las prácticas próximas que tienen reservadas. El objetivo es maximizar el uso de los recursos.
- Automatización del mantenimiento de las agendas mediante la planificación de la ejecución de los procesos batch.

Anexo A. Planificación y presupuestos

En este anexo se incluye toda la información relativa a la estimación de tiempos del proyecto y el presupuesto económico del mismo. La gestión de ambos conceptos durante la vida de un proyecto es crítica ya que las desviaciones en cualquiera de los dos apartados afectan bien a la fecha estimada de entrega o bien a la rentabilidad del mismo, pudiendo poner en riesgo la viabilidad del proyecto.

a. Planificación del desarrollo

A continuación se muestra el inventario de fases principales del proyecto y su esfuerzo en días laborables:

Nombre de la tarea	Días
Toma de requisitos	2
Análisis	5
Diseño	5
Implementación	32
Pruebas	5
Documentación	10
Fin de proyecto	1
TOTAL	**60**

Tabla 79: Esfuerzo

A la vista de los datos se puede apreciar que es un proyecto equilibrado en cuanto al reparto de los tiempos entre las diferentes fases.

La traslación del esfuerzo en una planificación lineal no siempre es directa ya que depende de los recursos implicados en cada fase y la posibilidad de solapar tareas.

En este caso, se ha dispuesto únicamente de un recurso, por lo que la relación esfuerzo-tiempo es directa. Asimismo, todas las fases se van a llevar a cabo de forma secuencial a excepción de la documentación que se realiza de forma simultánea al resto de fases.

A continuación se muestra la planificación del proyecto.

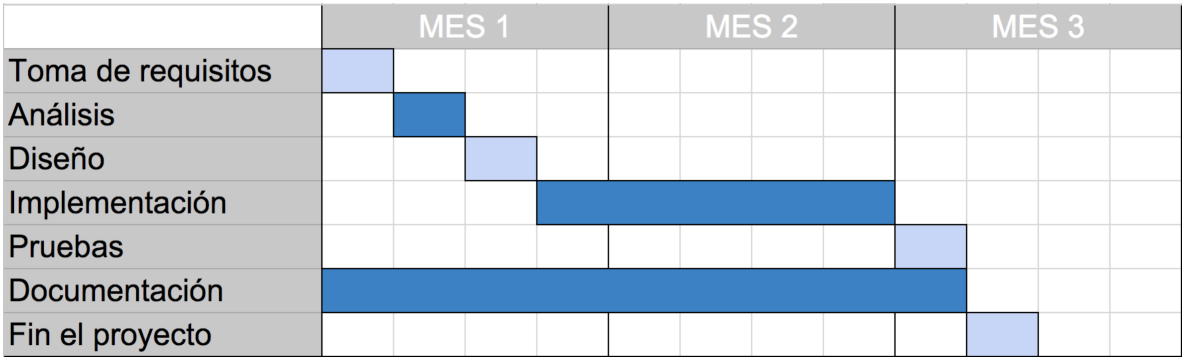


Figura 13: Planificación

b. Presupuesto

El presupuesto se divide en tres grandes conceptos:

- Personal: Que refleja el coste las personas que han intervenido en el proyecto.
- Software: Que refleja el coste de las licencias de las herramientas utilizadas durante el proyecto.
- Hardware: Que refleja el coste de los equipos físicos para el desarrollo del proyecto. En este caso no se incluye el coste de los servidores de puesta en marcha puesto que es un proyecto piloto.

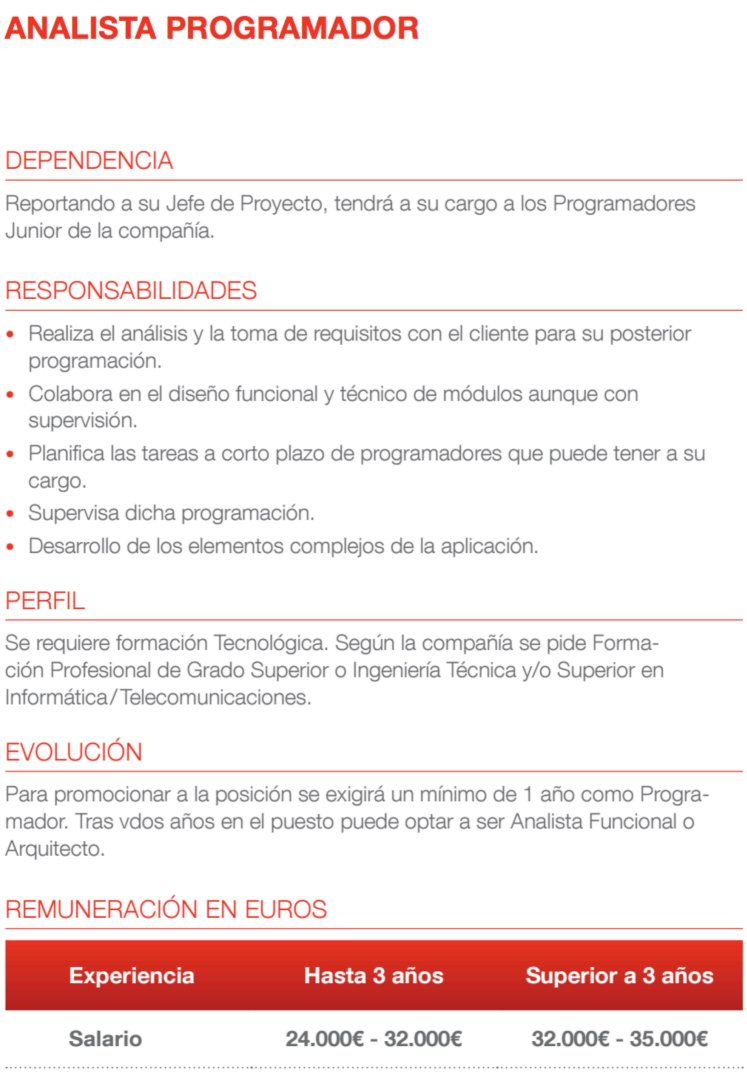
Personal

El proyecto tiene asignado un único recurso para todas las fases. Este recurso tiene el perfil de Analista programador de más de 3 años de experiencia.

Para saber el coste real del personal hay que contar, al menos, con los costes directos del mismo que son: salario bruto y seguridad social.

Como se indica en el siguiente artículo de BBVA (http://www.bbvacontuempresa.es/a/se-calculan-todos-los-costes-personal), los costes de la Seguridad Social que debe asumir la empresa al margen de su salario están de media en torno a un 33%. En este caso no se va a tener en cuenta los costes indirectos (como despidos, formación, etc.)

Según el estudio de remuneración de la consultora de recursos humanos Page Personnel (http://www.pagepersonnel.es/productsApp_pp_es/Estudios Remuneracion/er_tecnologia.pdf) esta es la descripción y la remuneración media de este perfil:



ANALISTA PROGRAMADOR

DEPENDENCIA

Reportando a su Jefe de Proyecto, tendrá a su cargo a los Programadores Junior de la compañía.

RESPONSABILIDADES

- Realiza el análisis y la toma de requisitos con el cliente para su posterior programación.
- Colabora en el diseño funcional y técnico de módulos aunque con supervisión.
- Planifica las tareas a corto plazo de programadores que puede tener a su cargo.
- Supervisa dicha programación.
- Desarrollo de los elementos complejos de la aplicación.

PERFIL

Se requiere formación Tecnológica. Según la compañía se pide Formación Profesional de Grado Superior o Ingeniería Técnica y/o Superior en Informática/Telecomunicaciones.

EVOLUCIÓN

Para promocionar a la posición se exigirá un mínimo de 1 año como Programador. Tras vdos años en el puesto puede optar a ser Analista Funcional o Arquitecto.

REMUNERACIÓN EN EUROS

Experiencia	Hasta 3 años	Superior a 3 años
Salario	24.000€ - 32.000€	32.000€ - 35.000€

Figura 14: Perfil Analista Programador (source: Page Personnel)

Atendiendo a la media de salarial, haremos los cálculos en base a un salario bruto de 33.500 €/brutos anuales. La fórmula sería:

(Sueldo bruto + Seguridad Social) % (Días laborables – Días de vacaciones)

Concepto	Valor
Sueldo bruto	33.500 €
Seguridad Social empresa (33%)	11.055 €
Días laborables 2015 (Madrid)	249
Vacaciones (días laborables)	22
TOTAL (coste por jornada)	**196 €**

Tabla 80: Cálculo coste jornada

El coste total del proyecto a nivel de personal es el resultado de multiplicar el número de jornadas (60) por el coste por jornada (196 €).

Por todo ello, el coste total de personal asciende a: 11.760 €

Software

Como se explica en el apartado de implementación, todas las herramientas utilizadas para el desarrollo del proyecto son gratuitas.

Asimismo, se ha utilizado una licencia de Microsoft Office 2011 para MAC que, al haber un acuerdo con los estudiantes de la Universidad Carlos III de Madrid, ha resultado igualmente gratuita

Por todo ello, el coste total de software asciende a: 0 €

Hardware

Como se ha indicado al principio de este anexo, este proyecto es un piloto de la solución definitiva, por lo que no se presupuestará el Hardware del entorno de explotación del mismo. Sin embargo, sí es necesario presupuestar el hardware utilizado para su desarrollo. En este caso ha sido el siguiente:

Concepto	Valor
Marca y modelo	Apple McBook Pro 15"
Versión	Intel Core i7 cuatro núcleos de 2,2 GHz
Memoria	256 GB almacenamiento flas PCIe
Precio oficial con IVA	**2.249 €**

Tabla 81: Precio MacBook Pro

El coste del hardware no se imputa de manera directa, sino que se imputa el resultado de la aplicación de la fórmula de amortización siguiente:

*(Meses uso * Coste * % dedicación) % Periodo de deprecación*

Dando como resultado lo siguiente:

Concepto	Valor
Meses de uso del hardware	2,5 meses
Coste del hardware con IVA	2.249 €
Dedicación	100%
Periodo de deprecación	60 meses
TOTAL	**93,7 €**

Tabla 82: Cálculo amortización

Por todo ello, el coste total de software asciende a: 93,7 €

Total

Como se ha indicado antes, el presupuesto total es la suma de los tres conceptos calculados anteriormente.

Concepto	Valor
Personal	11.760 €
Software	0 €
Hardware	93,7 €
Costes indirectos (10%)	**1.185,37 €**
TOTAL	**13.039,07 €**

Tabla 83: Cálculo presupuesto total

Por todo ello, el presupuesto total del proyecto asciende a: 13.039,07 €

Anexo B. Documentación del API

Este anexo recoge la documentación del API generada en RAML.

Toda la documentación recogida en este Anexo se puede consultar online y actualizada en:
https://anypoint.mulesoft.com/apiplatform/beeva/#/portals/organizations/75abf700-88a7-42d7-9bdd-ae42040809bb/apis/9989/versions/10289/pages/8276

Interfaz RAML

Este es el interfaz principal de la herramienta, donde se pueden ver de un simple vistazo todos los métodos incluidos así como sus métodos de acceso.

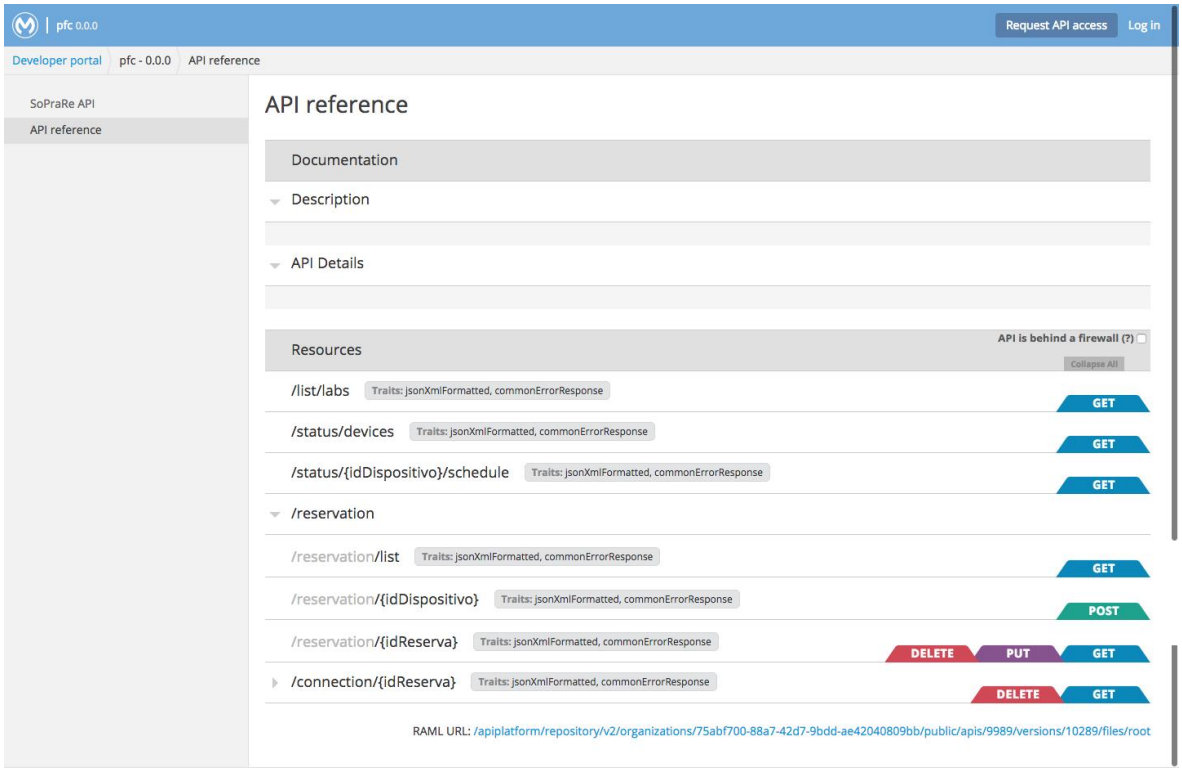


Figura 15: API - Resumen de métodos

/list/labs

Corresponde al método "Buscar laboratorios". Es un método tipo GET.

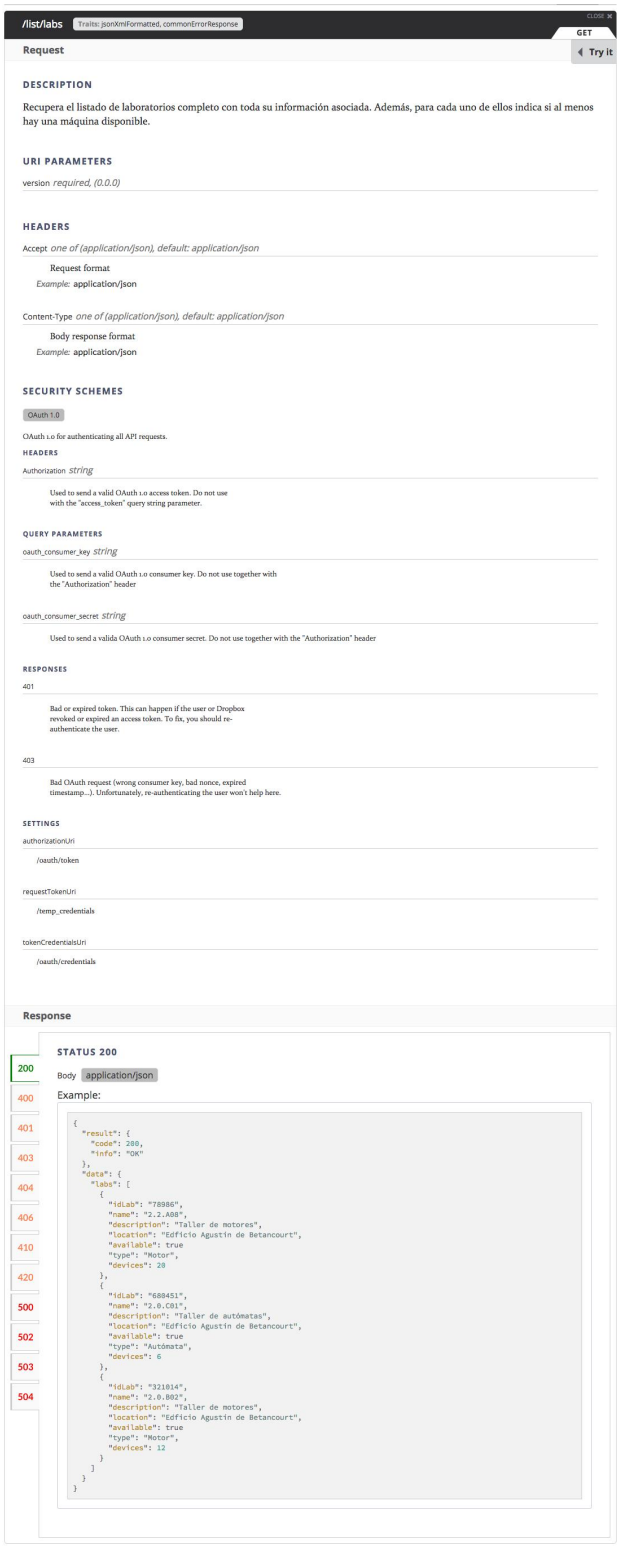


Figura 16: API - Buscar laboratorios

/status/devices

Corresponde al método “Buscar dispositivos”. Es un método tipo GET.

Figura 17: API - Buscar dispositivos

/status/{idDispositivo}/schedule

Corresponde al método "Recuperar agenda dispositivo". Es un método tipo GET.

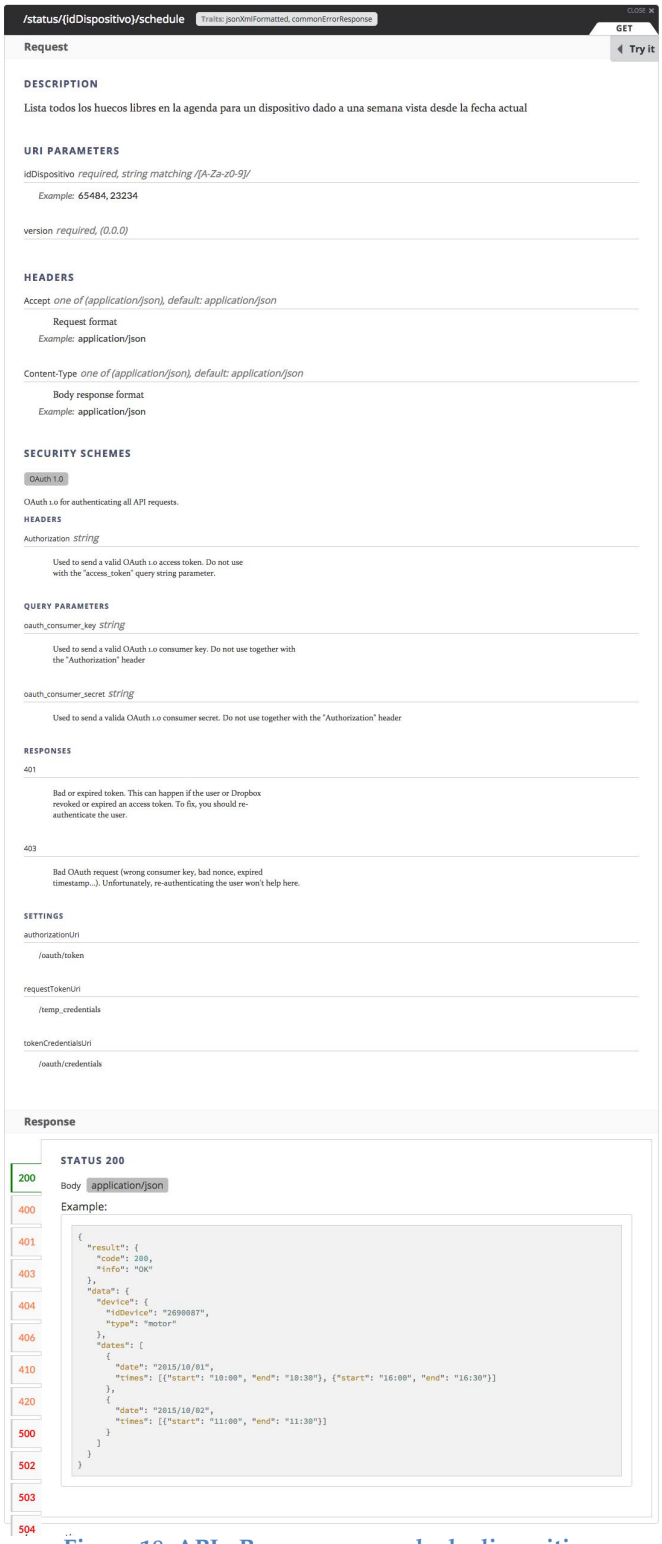


Figura 18: API - Recuperar agenda de dispositivo

/reservation/list

Corresponde al método "Recuperar reservas usuario". Es un método tipo GET.

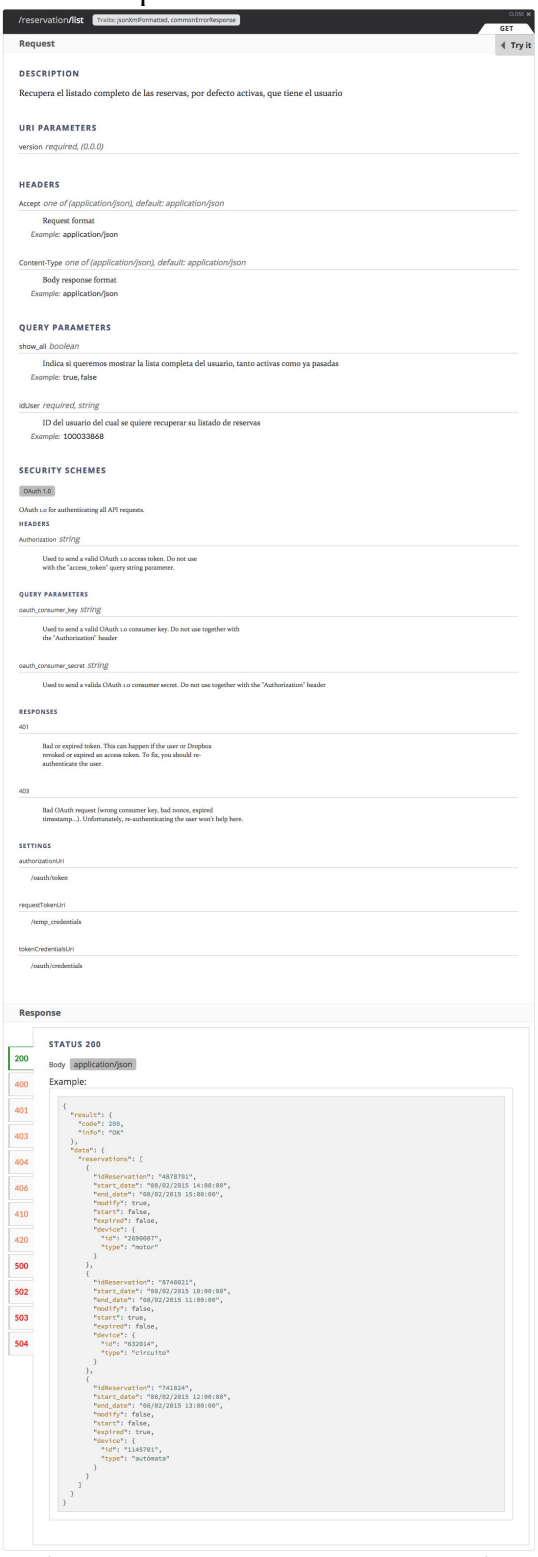


Figura 19: API - Recuperar reservas usuario

/reservation/{idDispositivo}

Corresponde al método "Crear nueva reserva". Es un método tipo POST.

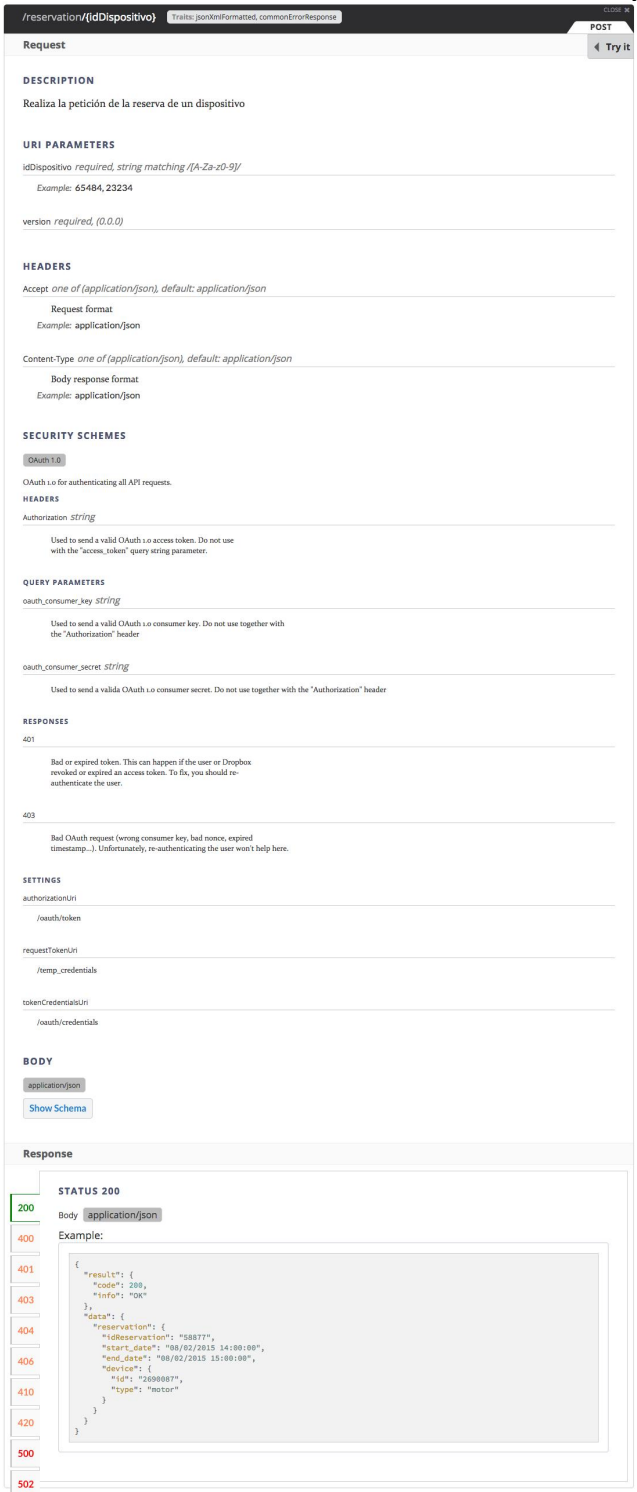


Figura 20: API - Crear nueva reserva

/reservation/{idReserva}

Tiene tres invocaciones diferentes.
El tipo "GET" corresponde al método "Recuperar detalle reserva".

Figura 21: API - Recuperar detalle reserva

El tipo “PUT” corresponde al método “Modificar reserva”.

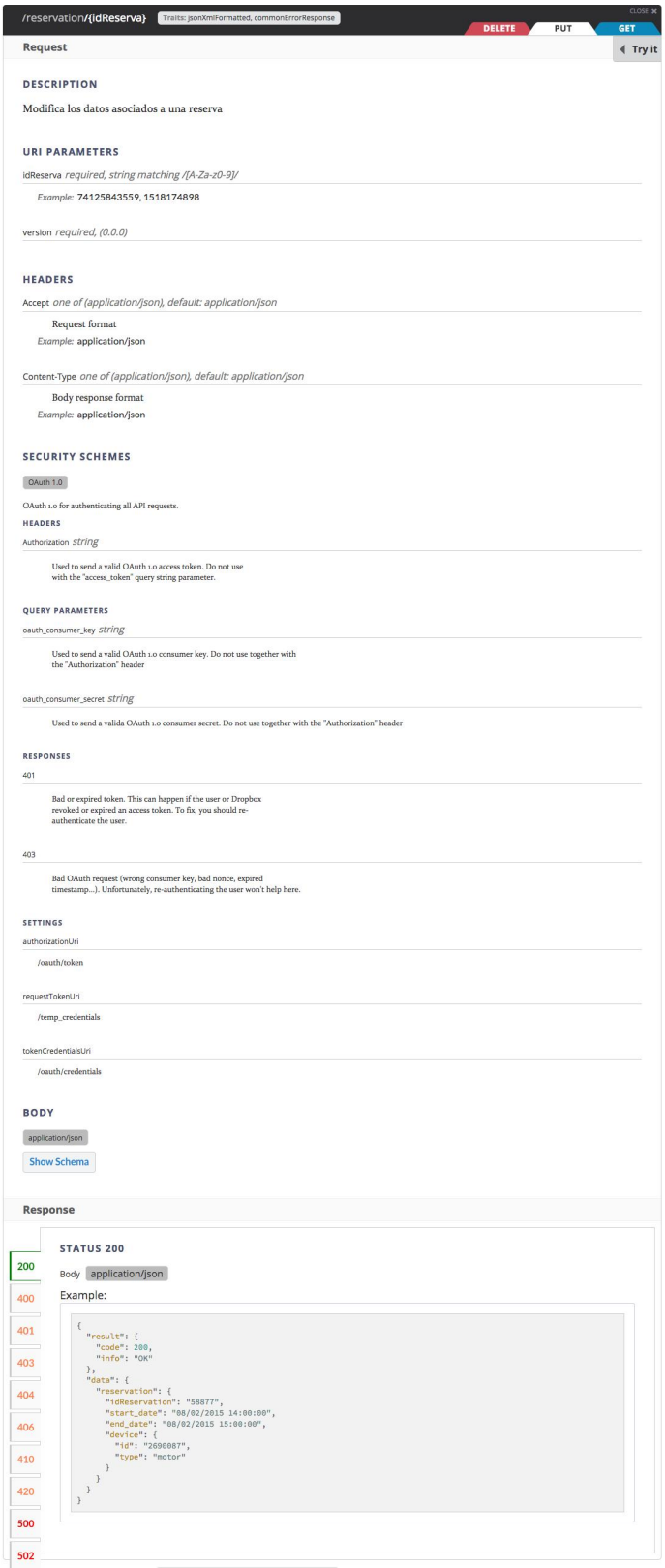


Figura 22: API - Modificar reserva

El tipo “DELETE” corresponde al método “Eliminar reserva”.

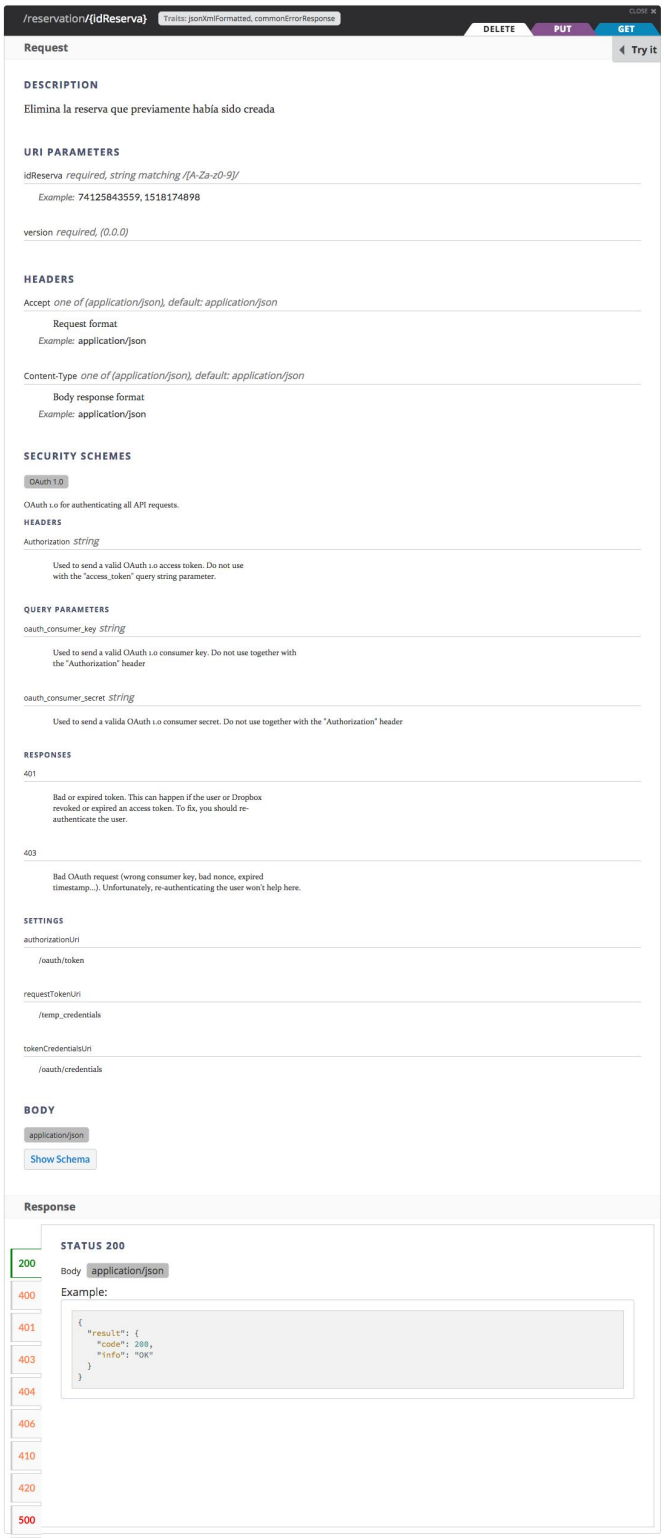


Figura 23: API - Eliminar reserva

/connection/{idReserva}

Tiene dos invocaciones diferentes.

El tipo "GET" corresponde al método "Abrir conexión".

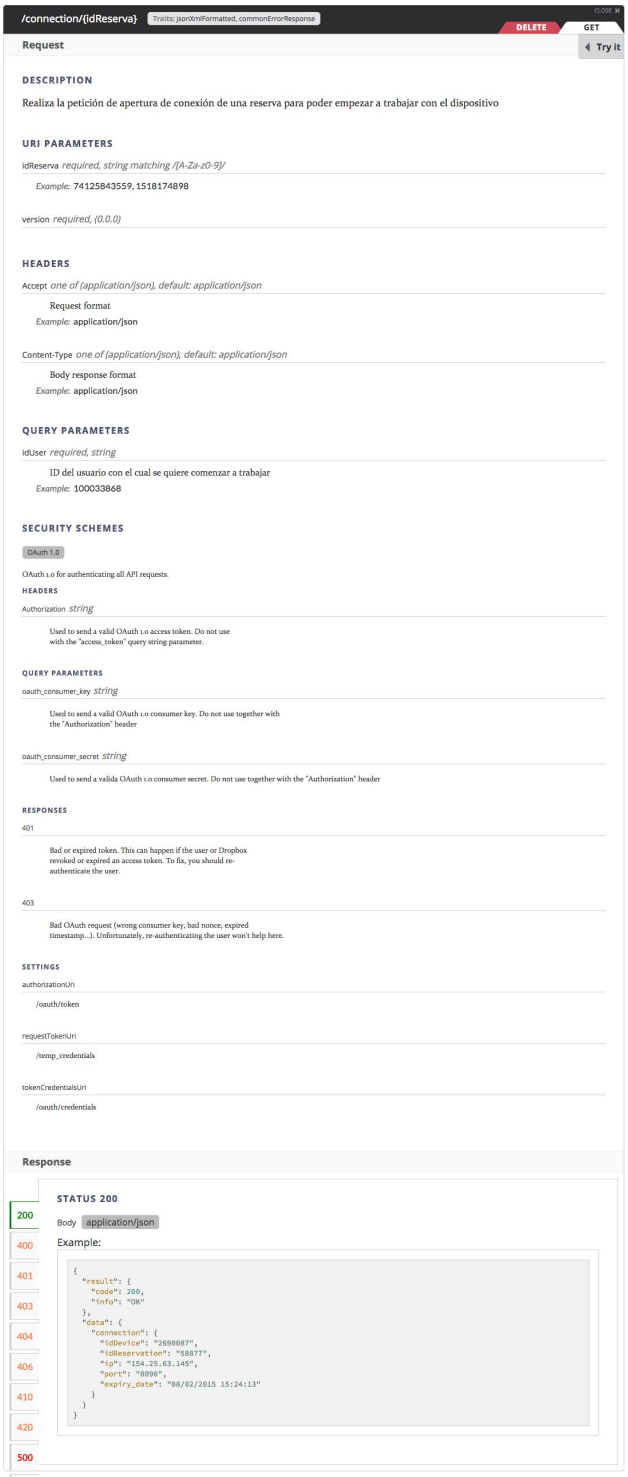


Figura 24: API - Abrir conexión

El tipo “DELETE” corresponde al método “Cerrar conexión”.

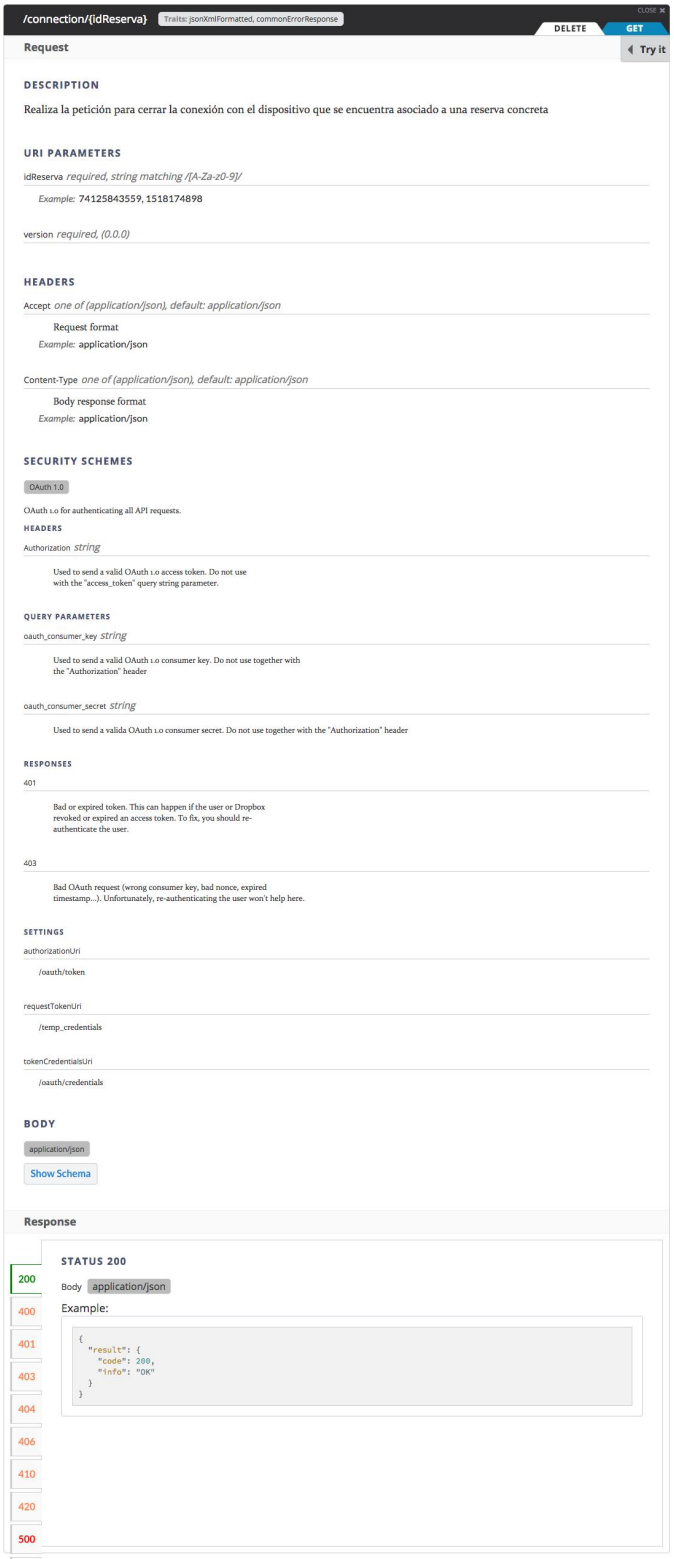


Figura 25: API - Cerrar conexión